CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Facultad de Informática

TRABAJO FIN DE GRADO

Propuesta de integración de agentes software inteligentes sobre JADE con entornos virtuales en Unity3D

Autor: Carlos Lozano Sánchez

Director: Angélica De Antonio Jiménez

MADRID, JUNIO DE 2013

A mis sobrinos Naya y Hugo por sus sonrisas

A mis padres y hermanos por su apoyo

A mis amigos por estar a mi lado

RESUMEN

La unión de distintos sistemas software constituye un elemento principal de las nuevas Tecnologías de la Información y la Comunicación. La integración de entornos virtuales tridimensionales con agentes software inteligentes es el objetivo que persigue este trabajo de investigación. Para llevar a cabo esta integración se parte de la creación de un agente virtual, un personaje que es controlado por una mente desarrollada siguiendo un enfoque basado en agentes software. Se busca así dotar al sistema de cierto nivel de inteligencia, tomando como referencia el funcionamiento del cerebro humano. Lo que se consigue es que el agente capte los estímulos del entorno, los procese y genere comportamientos, tanto reactivos como deliberativos, que son ejecutados por el personaje. Los resultados obtenidos resaltan el dinamismo del sistema, a la vez que animan a seguir investigando en este campo lleno de aplicaciones directas y reales sobre el mundo. En conclusión, esta investigación busca y consigue un nuevo paso en el progreso de las nuevas tecnologías mediante una integración real de distintos sistemas software.

ABSTRACT

The union of different software systems is a major element of Information and Communications Technology. The aim of this research is the integration of 3D virtual environments and intelligent software agents. This integration is based on the development of a virtual agent, a character that is controlled by a mind developed following a software agent approach. It is sought to provide the system with some intelligence level, taking the human brain function as a reference point. The consequence is that the agent captures environmental stimuli, processes them and creates reactive and deliberative behaviours that can be executed by the avatar. The findings emphasize the dynamism of the system as well as they encourage to research more in this field that has a lot of direct and real-life applications on the world. In conclusion, this research seeks and takes a new step in the progress of new technologies through a real integration of different software systems.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1.- INTRODUCCIÓN

La ciencia y la tecnología han sufrido un gran avance durante los siglos XIX y XX, provocado por la Revolución Industrial, las Guerras Mundiales y la Guerra Fría, entre otros eventos históricos. No menos importante es el cambio social, cultural y económico producido durante estos siglos. Todos estos aspectos son los que han modelado y definido la sociedad actual. Hoy en día, principios de siglo XXI, nos encontramos sumidos en una crisis mundial donde las Tecnologías de la Información y la Comunicación están jugando un papel fundamental en todos los cambios que estamos experimentando. Si comparamos el mundo que nos rodea con el de hace una década, nos damos cuenta que ha cambiado, desde la forma de comunicarnos o socializar hasta el funcionamiento de la economía mundial, pasando por todos los aspectos de la sociedad.

Este proyecto tiene como objetivo general, apostar por la integración tecnológica como elemento imprescindible para el progreso y desarrollo de la sociedad global en la que vivimos. En particular, se apuesta por la integración de agentes software y entornos virtuales para explorar nuevas posibilidades de aplicar las Tecnologías de la Información y las Comunicaciones a la enseñanza y el aprendizaje.

El trabajo surge de los proyectos Virtual Reality for Inspection, Maintenance, Operation and Repair of Nuclear Power Plants (VRIMOR) y Modelos de Interacción centrados en Lenguaje, Espacio y Semántica computacional (MILES). En estos proyectos se ha observado la necesidad de tener un sistema que permita el desarrollo ágil de comportamientos de avatares y agentes virtuales autónomos en entornos virtuales. Para resolver el problema se propone, como se ha mencionado anteriormente, un enfoque basado en agentes software que se irá desarrollando en los capítulos posteriores.

El capítulo 2 ofrece una visión global de los principales campos que sustentan el proyecto, así como una descripción de VRIMOR y MILES.

El capítulo 3 trata de forma detallada y organizada el trabajo realizado. Comienza con una descripción precisa del proyecto. A continuación, se explica cómo ha sido planificado y desarrollado. Seguidamente, se plantea la arquitectura genérica, la concreción contextual y el diseño detallado del sistema. Finalmente, se especifican las pruebas realizadas.

El capítulo 4 expone los resultados y las conclusiones obtenidas, así como las posibles líneas futuras de trabajo.

2.- ESTADO DE LA CUESTIÓN

Como se ha mencionado en la Introducción, el proyecto va a integrar agentes software y entornos virtuales con el objetivo de dar soporte a la creación de agentes virtuales inteligentes. Por ello, en primer lugar y para comprender el área del proyecto vamos a definir qué es integración, qué son los agentes software, qué son los entornos virtuales y qué son los agentes virtuales.

La integración de soluciones software hace referencia a la unión de distintos sistemas independientes, que utilizan o no la misma tecnología, para conseguir una mayor funcionalidad, modularidad y extensibilidad a menor coste. Para obtener una idea de lo importante que es la integración únicamente hay que observar que la mayoría de los servicios web 2.0 se integran entre ellos para conseguir ser más completos y ofrecer mayor funcionalidad a sus usuarios.

Respecto a los agentes software, muchos autores han intentado definirlos con mayor o menor acierto. (Wooldridge y Jennings, 1998) definen los agentes como una herramienta de abstracción para el desarrollo de sistemas complejos y distribuidos. Además (Wooldridge y Jennings, 1995) describen una serie de características fundamentales de los agentes:

- Autonomía: los agentes operan sin la intervención directa de humanos u otros, y tienen algún tipo de control acerca de sus acciones y estado interno.
- Capacidad social: los agentes interactúan con otros agentes (y posiblemente humanos) por medio de algún tipo de lenguaje de comunicación de agentes.
- Reactividad: los agentes perciben su entorno, y responden de un modo oportuno a los cambios que en él suceden.
- Proactividad: los agentes no actúan simplemente como respuesta al entorno, sino que son capaces de exhibir un comportamiento dirigido por metas, tomando la iniciativa.

Franklin y Graesser identifican también las siguientes características, aparte de las citadas:

- Continuidad temporal: los agentes son ejecutados de forma continua.
- Adaptación o aprendizaje: los agentes, a partir de su experiencia previa, evolucionan ofreciendo un comportamiento adaptado al entorno.
- Movilidad: los agentes son capaces de transportarse a sí mismos de una máquina a otra según las necesidades del momento.
- Flexibilidad: las acciones de los agentes no están descritas con anterioridad sino que dependen de la situación del entorno actual.
- Caracterización: los agentes pueden presentar personalidades y estados emocionales.

Como puede observarse, es posible que un agente tenga muchas características pero no es necesario que presente todas, puesto que dependerá de las necesidades del sistema que se está desarrollando.

Por otra parte, un entorno virtual es una representación de la realidad a través de su simulación. A pesar de la multitud de tipos de entornos virtuales que existen, el proyecto únicamente se centra en entornos virtuales 3D que intentan simular un espacio tridimensional real o ficticio, como puede ser un edificio, una ciudad o una nave espacial, entre otros ejemplos.

Por último, es necesario definir qué es un agente virtual para comprender el proyecto en su totalidad. Actualmente, existen muchos tipos de agentes virtuales como, por ejemplo, los *Non-Player Characters* de Second Life o los robots telefónicos de atención al cliente. Todos ellos tienen como objetivo común actuar de manera autónoma en su entorno con un determinado fin (ayudar a otro jugador, ayudar a un cliente, simular un protocolo de emergencias,…). En este proyecto un agente virtual es un personaje, con comportamientos autónomos e inteligentes, que se encuentra en un entorno virtual. La mayor parte de la investigación actual, relativa a este tipo de agentes, se ha centrado en cómo ofrecer un comportamiento humano a los avatares con forma humanoide. Por el contrario, el actual proyecto busca generalizar los comportamientos con independencia de la estructura corporal del avatar.

A continuación son presentados en detalle los proyectos VRIMOR y MILES, precursores del presente trabajo, para comprender su influencia en el actual.

VRIMOR pretendía elaborar un conjunto de herramientas de simulación por ordenador para la planificación, entrenamiento y evaluación de las labores de mantenimiento de una central nuclear. Los principales objetivos eran reducir los costes de formación y aumentar la seguridad mediante el uso de estas herramientas. Para alcanzar el mayor realismo posible se realizó un entorno virtual con la misma apariencia y distribución que la central nuclear. Para realizar la planificación de una intervención en la central nuclear, el entorno ofrecía un avatar, con comportamientos básicos, que era controlado por el usuario mediante órdenes del tipo “agáchate”, “coge el martillo” o “camina hasta la escalera”. El presente proyecto parte de la misma idea y ha sido complementado con comportamientos autónomos.

Por su parte, MILES es un proyecto en actual desarrollo por el Laboratorio Decoroso Crespo en la Facultad de Informática de la Universidad Politécnica de Madrid. Su finalidad es guiar, mediante lenguaje natural, a usuarios dentro de entornos virtuales, teniendo en cuenta sus preferencias, conocimientos previos, necesidades y discapacidades. Un ejemplo ilustrativo que nos ayuda a comprender el sistema es el siguiente:

Imaginemos que eres un alumno nuevo de la facultad, te encuentras en la entrada del bloque 5 y tienes que ir a una clase del bloque 6. Si partimos de estas premisas, el sistema te iría guiando, a cada instante, haciendo referencia a objetos visibles (ascensores, extintores, tablones de anuncios, puertas, entre otros) e indicando si tienes que subir, bajar escaleras o utilizar el ascensor dependiendo de tus preferencias definidas, discapacidad o necesidad. Ahora bien, si eres un alumno con cierta antigüedad, el sistema te guiará de forma menos detallada y haciendo referencia a lugares conocidos por todas las personas, que trabajan o estudian en la facultad, o por ti en particular.

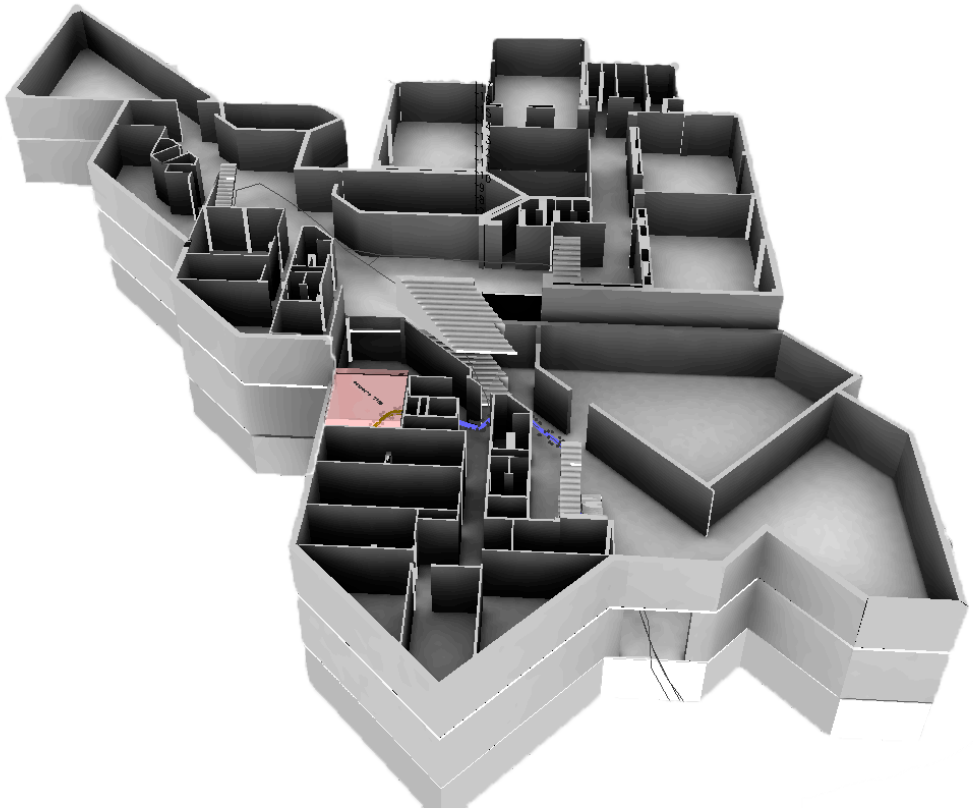

Figura 1: Bloque 5 y 6 de la Facultad de Informática

Como se puede observar, MILES es un sistema complejo que pretende adaptarse a los usuarios, aprender de ellos y guiarlos mediante lenguaje natural. Para lograr su objetivo utiliza una plataforma multiagentes, procedente de un proyecto anterior denominado MAEVIF, que proporciona la inteligencia artificial necesaria. Entre los agentes de la plataforma se puede destacar el Agente Comunicación, el Agente Trayectoria y el Agente Mundo, que serán reutilizados por el presente proyecto para la generación de ciertos comportamientos y la comunicación con el entorno virtual.

3.- DESARROLLO

3.1.- Componentes del Proyecto

El Agente Virtual es un agente software que es representado, mediante un avatar, en un entorno virtual tridimensional. Su funcionamiento es sencillo:

1. El avatar percibe el entorno y envía las percepciones al agente software.
2. El agente software procesa las percepciones, determina comportamientos adecuados y envía dichos comportamientos al avatar.
3. El avatar ejecuta los comportamientos cambiando el entorno.

Superficialmente, el funcionamiento puede compararse al de las personas. El avatar sería el cuerpo, el agente software actuaría como el cerebro y la comunicación se realizaría mediante el sistema nervioso.

A continuación, para comprender mejor el proyecto, se describe brevemente cada una de sus partes.

El primer componente es un entorno virtual tridimensional realizado con Unity3D, un motor de videojuegos. El escenario elegido como ejemplo es un edificio de tres plantas con dieciséis habitaciones, escaleras, ascensores, puertas, objetos decorativos, etc.

La razón principal de elegir un escenario de este tipo ha sido la gran variedad de comportamientos que se pueden desarrollar en él, demostrando el potencial que se puede obtener con este Agente Virtual en entornos virtuales tridimensionales.

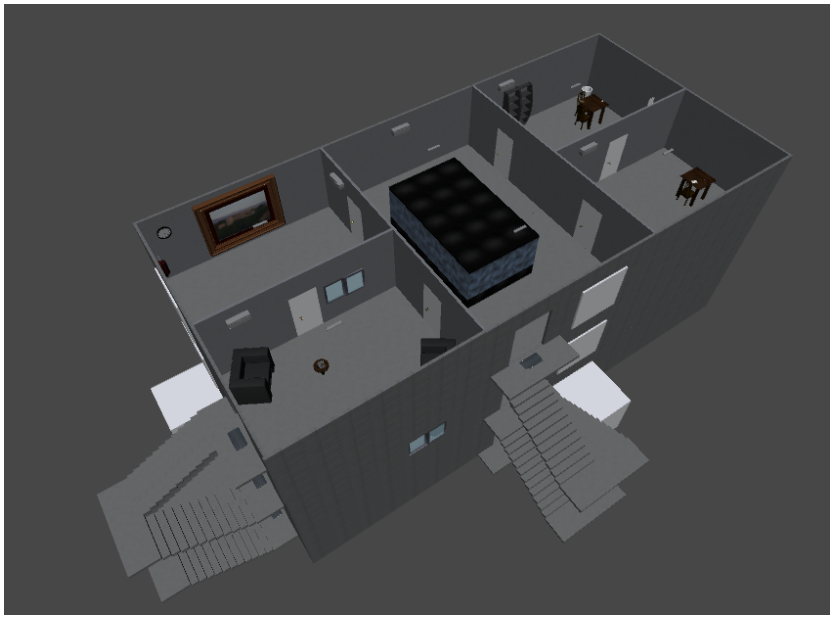

Figura 2: Entorno Virtual Tridimensional

El segundo componente es el avatar, es decir, la representación del Agente Virtual en el escenario. La labor del avatar es percibir los cambios del entorno y ejecutar los comportamientos que le ordena la mente del Agente Virtual. La estructura corporal dependerá de la concreción contextual realizada (Ver apartado 3.5.- Concreción Contextual). En este proyecto se ha decidido que el avatar sea un robot arácnido (modelado y animado por terceros) que nos proporciona una variedad amplia de animaciones para la visualización de los distintos comportamientos.

Figura 3: Avatar del Agente Virtual

El tercer componente, y más importante, es el Agente Virtual realizado con JADE, un framework para el desarrollo de agentes software en Java, e integrado dentro de la plataforma multiagentes de MILES. Se encarga de procesar las percepciones que envía el avatar y generar comportamientos que el avatar pueda ejecutar. El agente presenta las siguientes características:

- Actúa de forma autónoma en la mayoría de los comportamientos, aunque para aumentar la funcionalidad se ha incluido un tipo de comportamiento semiautónomo o dirigido por órdenes del usuario.
- Es social, ya que interactúa con otros agentes de la plataforma MILES para planificar ciertos comportamientos (Agente Mundo y Agente Trayectoria) y comunicarse con el entorno virtual y el controlador (Agente Comunicación).
- Es reactivo porque responde inmediatamente a determinadas percepciones.
- Es proactivo puesto que ciertos comportamientos son completamente deliberativos o guiados por metas.
- Presenta continuidad temporal, es decir, se ejecuta de forma continua llevando a cabo su labor.
- Se adapta, de forma limitada, al entorno según su experiencia previa.

En el apartado 3.5.- Concreción Contextual se detalla qué percepciones y comportamientos se han definido e implementado en el Agente Virtual para probar su funcionamiento.

El cuarto y último componente del sistema es el Controlador, una aplicación independiente desarrollada en Java, que permite al usuario enviar órdenes al Agente Virtual para que realice ciertos comportamientos semiautónomos.

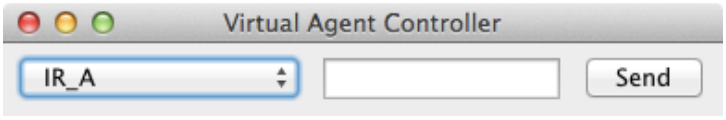

Figura 4: Interfaz Gráfica del Controlador del Agente Virtual

Resumiendo, el proyecto está compuesto por el Cliente Gráfico (escenario tridimensional más avatar), el Agente Virtual (agente software) y el Controlador (aplicación para enviar órdenes).

3.2.- Percepciones y Comportamientos

En el apartado anterior se ha realizado una descripción superficial tanto del funcionamiento como de los componentes del sistema. A continuación, se ofrece una visión más formal del funcionamiento.

El Agente Virtual puede ser representado como una caja negra que recibe percepciones y genera comportamientos.

Figura 5: Esquema del Agente Virtual

3.2.1.- Percepciones

Las percepciones son todos aquellos estímulos que recibe el Agente Virtual y mediante los cuales forma su propia representación de la realidad del entorno. Se han concretado tres tipos distintos de percepciones:

- **Percepciones del mundo.** Son todos los estímulos que percibe el avatar en el entorno virtual.
- **Percepciones internas.** Son aquellos estímulos que percibe internamente el propio Agente Virtual.
- **Órdenes del usuario.** Son estímulos enviados por el usuario al Agente Virtual.

Cada estímulo tiene una precondición necesaria para poder ser percibido por el agente. Por ejemplo, no se podrá percibir a otro avatar del entorno virtual si no se encuentra dentro del campo de visión del avatar del Agente Virtual.

3.2.2.- Comportamientos

Los comportamientos son las maneras de actuar del Agente Virtual ante los estímulos percibidos. Se ha tenido en cuenta tres tipos distintos de comportamientos:

- **Comportamientos reactivos.** Surgen de forma inmediata e inconsciente ante un cambio en el entorno.
- **Comportamientos deliberativos.** Surgen de forma autónoma y consciente con el objetivo de alcanzar una meta, no fijada por el usuario, a través de la planificación de determinadas acciones.
- **Comportamientos semiautónomos.** Surgen de forma semiautónoma y consciente con el objetivo de alcanzar una meta, fijada por el usuario, a través de la planificación de determinadas acciones.

Resumiendo, un comportamiento reactivo genera normalmente una acción simple (reacción), mientras que los deliberativos y semiautónomos generan mayoritariamente acciones compuestas (planes).

Para disminuir la complejidad del sistema se han tomado las siguientes decisiones de diseño:

1. El sistema crea y gestiona tareas. Una tarea es una acción atómica (indivisible) que el avatar puede ejecutar.
2. Todos los comportamientos están compuestos por una o varias tareas. Por ejemplo, si el comportamiento es abrir una puerta de la habitación donde se encuentra el avatar, se subdivide en dos tareas: (1) ir hasta la posición de la puerta y (2) abrir la puerta.
3. El sistema solamente maneja un plan al mismo tiempo.
4. Tenemos que poder distinguir las tareas que forman parte de planes de las reactivas.
5. Se permite definir planes que pueden ser cancelados si llega un plan nuevo. Esto conlleva tener la capacidad de poder eliminar todas las tareas creadas por el plan anterior para poder añadir las tareas del nuevo plan, de manera que se cumpla la restricción número 3.
6. Las tareas creadas por planes necesitan ser confirmadas cuando se ha completado su ejecución, debido a que pueden extenderse en el tiempo. Por ejemplo, ir a una posición lleva un tiempo de desplazamiento variable dependiendo de la distancia y la velocidad de movimiento.

3.3.- Planificación y Seguimiento

3.3.1.- Planificación

Para el desarrollo de este proyecto, con la funcionalidad planteada en los apartados anteriores, se ha necesitado una planificación realista y precisa. Se disponía de cuatro meses para desarrollar la totalidad del sistema y cumplir con los siguientes objetivos iniciales del proyecto:

1. Definir una interfaz de comunicación entre Unity3D y Jade que permita el envío de información, acerca del entorno, al agente software y el control, por parte de éste, del comportamiento del avatar que le representa en el entorno 3D.
2. Establecer una serie de comportamientos básicos para el agente y programar tanto la lógica en Jade como su realización en Unity3D, incluyendo al menos la capacidad de desplazarse por el entorno 3D y de interactuar con los usuarios.
3. Desarrollar un agente de prueba en el entorno 3D.
4. Realizar pruebas de funcionamiento.
5. Documentar el proceso de desarrollo.
6. Documentar los mecanismos para la creación de nuevos agentes virtuales sobre la base de lo desarrollado.

Como primer paso, se analizaron las necesidades del proyecto, lo que permitió la definición del conjunto de tareas, que se especifican a continuación, y que debían llevarse a cabo para completar toda la funcionalidad:

1. Definición de la arquitectura.
2. Configuración de los entornos de desarrollo.
3. Desarrollo del agente dentro de la plataforma de agentes de MILES.
   3.1. Definición de los comportamientos.
   3.2. Planificación de los comportamientos.
   3.3. Implementación del agente utilizando JADE.
   3.4. Integración con otros agentes de la plataforma.
4. Realización del escenario tridimensional en Unity3D.
   4.1. Modelado del escenario.
   4.2. Decoración del escenario.
   4.3. Percepción de los cambios del mundo.
   4.4. Integración de los comportamientos básicos con las animaciones del avatar.
   4.5. Desarrollo del proceso de interacción entre los objetos del escenario.
5. Integración de la plataforma Unity3D con la plataforma de agentes JADE.
   5.1. Definición del protocolo de comunicación.
   5.2. Definición de los mensajes.
   5.3. Implementación de la comunicación entre las dos plataformas.

6. Realización de pruebas.
7. Documentación.

Una vez establecidas las tareas, se distribuyeron en siete etapas durante los cuatro meses del desarrollo:

1. La primera etapa se enfocó al aprendizaje de las herramientas y a la configuración correcta de los entornos utilizados.
2. En la segunda etapa, se consiguió la definición de la arquitectura del sistema y del conjunto de comportamientos que se iban a desarrollar.
3. La tercera, se centró en la realización de la integración y comunicación entre el entorno virtual y la plataforma de agentes.
4. Las siguientes tres etapas corresponden a cada una de las tres fases de desarrollo del agente. En cada una de ellas se implementa un tipo de percepción y se realizan las pruebas de las percepciones incorporadas.
5. La última etapa, se dedicó a la depuración del sistema y a la formalización de la documentación del proyecto.

A continuación, se muestra gráficamente la distribución de las tareas mediante un diagrama de Gantt:

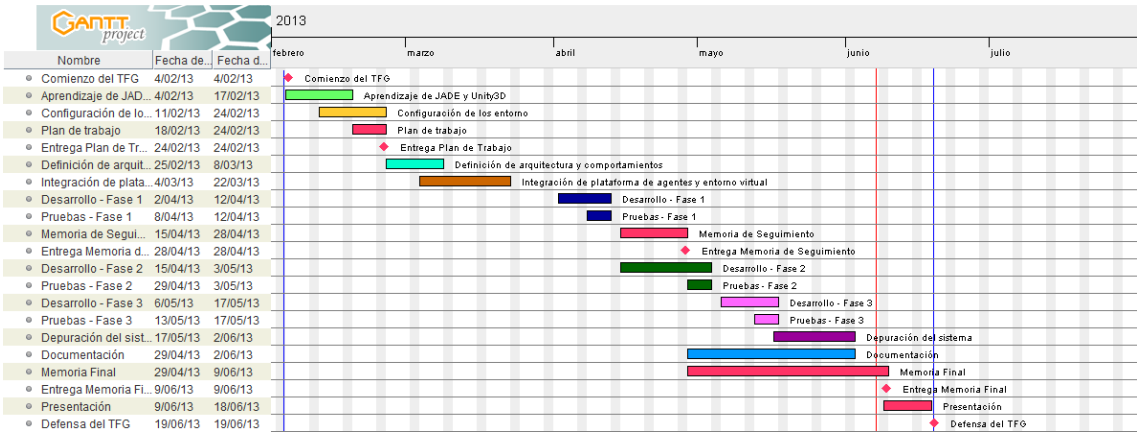

Figura 6: Diagrama de Gantt

3.3.2.- Seguimiento

El proyecto ha sido ejecutado según la planificación definida en el apartado anterior. Para completar todas y cada una de las fases, en los cuatro meses de desarrollo, ha sido necesario emplear 480 horas, es decir, 3 personas-mes.

3.4.- Arquitectura

En el apartado 3.1.- Componentes del Proyecto se ofrece una vista genérica del proyecto con sus componentes y piezas clave. Por su parte, este apartado ofrece una descripción detallada de los componentes y la arquitectura del sistema, siguiendo un enfoque desde lo más genérico a lo más específico. El diseño de la arquitectura ha sido esencial en el proyecto para conseguir un sistema organizado, modular y extensible.

3.4.1.- Arquitectura del Sistema

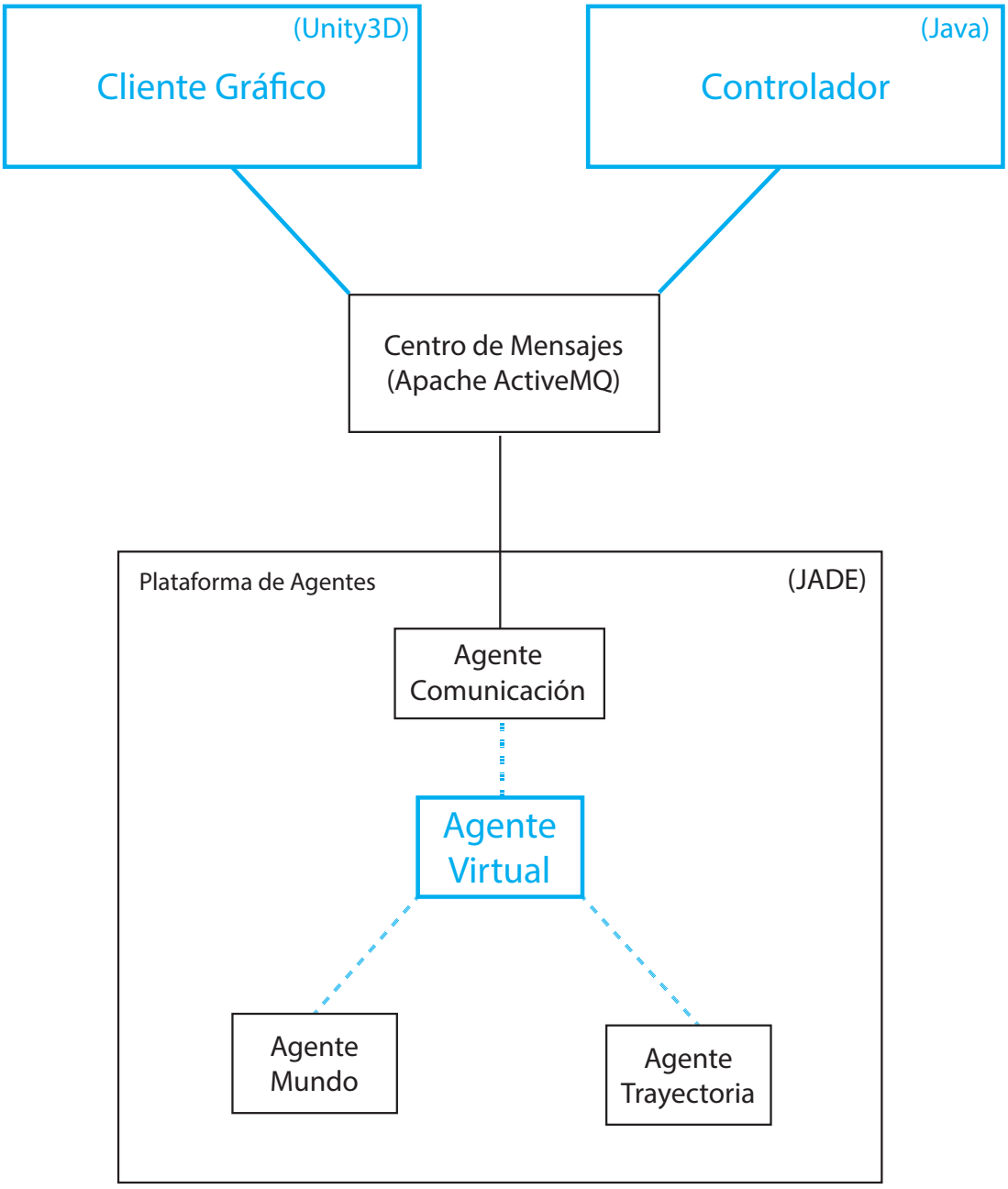

Figura 7: Arquitectura del Sistema

Como puede observarse, el sistema se compone de cuatro elementos:

1. **Cliente Gráfico.** Es la capa de visualización del sistema que contiene el escenario tridimensional y el avatar. Además, es la encargada de capturar las percepciones del mundo y de ejecutar los comportamientos.
2. **Controlador.** Es la aplicación que permite al usuario enviar órdenes al Agente Virtual para que realice ciertos comportamientos semiautónomos.
3. **Centro de Mensajes.** Es un servidor de mensajería que permite la comunicación entre los distintos componentes y tecnologías.
4. **Plataforma de Agentes.** Es un conjunto de agentes software que colaboran entre sí para alcanzar un objetivo común. El Agente Virtual forma parte de dicha plataforma y se encuentra integrado con los siguientes agentes del proyecto MILES:
    a. **Agente Comunicación.** Se encarga de abstraer la comunicación de los agentes con el exterior a través del centro de mensajes.
    b. **Agente Mundo.** Permite representar y consultar el estado actual del mundo apoyándose en una ontología.
    c. **Agente Trayectoria.** Calcula trayectorias óptimas entre distintos puntos del escenario. La trayectoria nos proporcionará datos relativos al punto inicial, las coordenadas de las puertas por las que hay que ir pasando y el punto final.

Los componentes de la arquitectura señalados en azul han sido desarrollados completamente en este proyecto. Las líneas azules continuas hacen referencia a las integraciones que han sido realizadas desde cero. Mientras que las líneas azules discontinuas son integraciones realizadas teniendo en cuenta el funcionamiento de los agentes afectados y manteniendo la compatibilidad con el proyecto MILES.

En los siguientes apartados se detallan las arquitecturas de los módulos desarrollados en este proyecto: Cliente Gráfico, Controlador y Agente Virtual.

3.4.2.- Arquitectura del Cliente Gráfico

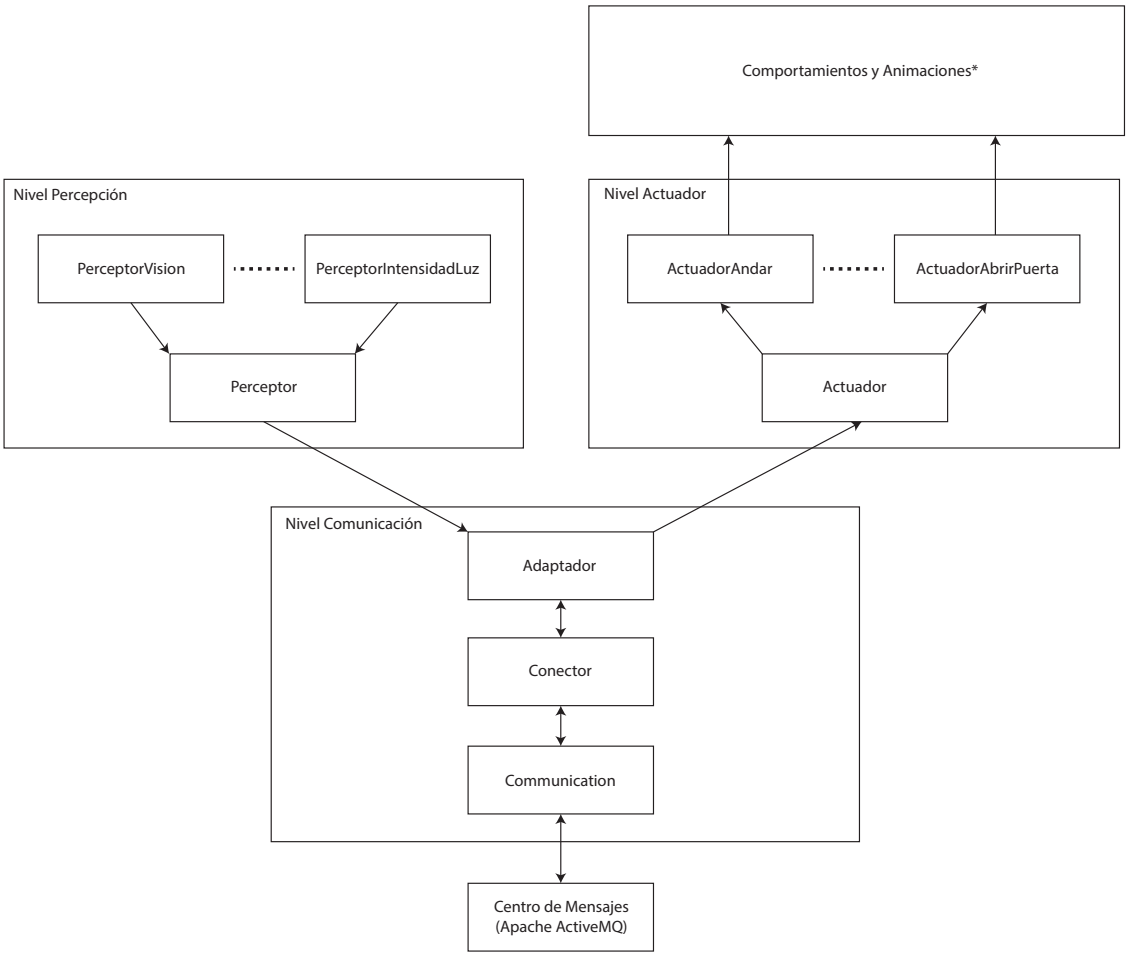

Figura 8: Arquitectura del Cliente Gráfico

El cliente gráfico se ha desarrollado con la herramienta Unity3D. Este motor de videojuegos multiplataforma permite ensamblar escenarios tridimensionales mediante objetos modelados y programar la capa lógica a través de scripts en C#, JavaScript o Boo.

En esta arquitectura se observa únicamente la capa lógica relacionada con el Agente Virtual. Todos los módulos han sido desarrollados por este proyecto a excepción de "Comportamientos y animaciones" que pertenece al robot arácnido. Además, se han programado otros scripts necesarios para completar la experiencia de usuario, como por ejemplo, un simulador de luz natural, las animaciones de las puertas o el control de las cámaras.

El nivel de comunicación abstrae la conexión y comunicación con el centro de mensajes a través de una interfaz. Los módulos de este nivel son:

- **Communication.** Es una librería para Java y C# que abstrae la tecnología *Java Message Service* permitiendo el envío y recepción de mensajes de manera más sencilla y transparente con el centro de mensajes.
- **Conector.** Se encarga de abstraer la localización del centro de mensajes y la tecnología utilizada para el envío y recepción de la información. Esto permite que en un futuro se pueda cambiar la tecnología de comunicación sin necesidad de realizar cambios en la capa lógica del sistema.
- **Adaptador.** Es el intermediario entre el conector y la capa lógica. Se encarga de establecer los parámetros de conexión con la plataforma de agentes (número de identificación del cliente, idioma, escenario,…), de *serializar* la información para su envío a través de la red y de *deserializar* la recibida para su uso por los niveles superiores.

El nivel de percepción se encarga de percibir todos los estímulos y cambios producidos en el entorno. Los módulos de este nivel son:

- **Perceptores específicos.** Conjunto de sensores especializados en detectar, cada uno, un tipo de estímulo. Un ejemplo de perceptor específico sería el perceptor de avatares que captaría todos los avatares dentro del campo de visión establecido.
- **Perceptor.** Con los datos obtenidos por los perceptores específicos, este módulo crea las percepciones para poder ser enviadas al Agente Virtual mediante el centro de mensajes.

Por último, el nivel de actuación es el responsable de que el avatar ejecute las tareas procedentes del Agente Virtual. Los módulos de este nivel son:

- **Actuador.** Recibe las tareas que el avatar tiene que ejecutar. Si son tareas simples llama directamente a la animación que corresponda del avatar. Si por el contrario, son complejas, delega la responsabilidad en un actuador específico.
- **Actuadores específicos.** Son los responsables de que el avatar ejecute tareas complejas que requieren de varias animaciones o de cálculos adicionales como, entre otros, calcular la trayectoria entre dos puntos dentro de una misma habitación evitando los obstáculos que haya.

3.4.3.- Arquitectura del Controlador

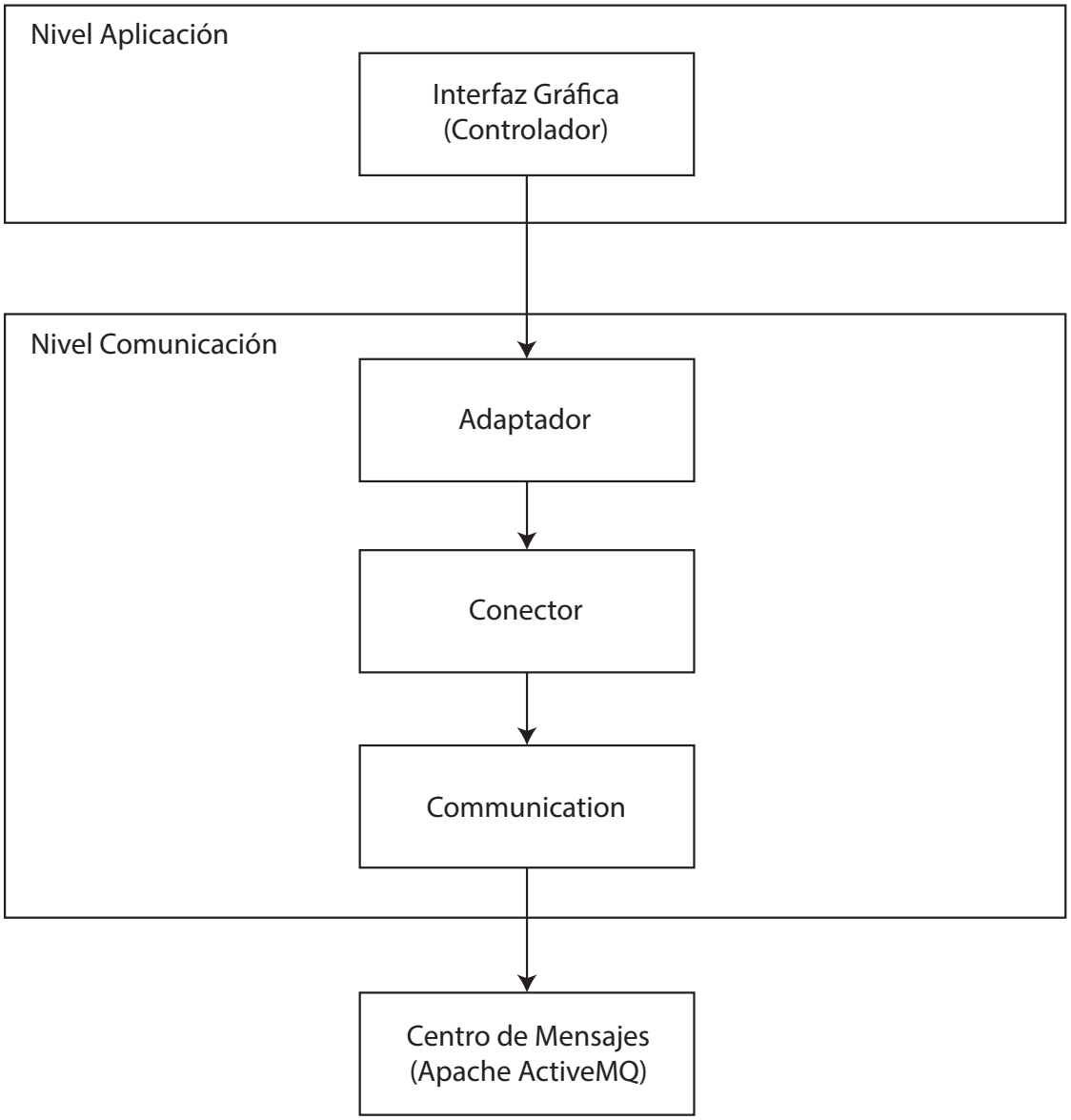

Figura 9: Arquitectura del Controlador

El Controlador es una aplicación Java muy sencilla que permite enviar órdenes del usuario al Agente Virtual.

La arquitectura de la aplicación está compuesta por dos niveles. El nivel de aplicación que permite la interacción del usuario para enviar sus órdenes (Ver Figura 4: Interfaz Gráfica del Controlador del Agente Virtual), y el nivel de comunicación que realiza el envío de las órdenes. El nivel de comunicación funciona exactamente de la misma manera que el del Cliente Gráfico.

3.4.4.- Arquitectura del Agente Virtual

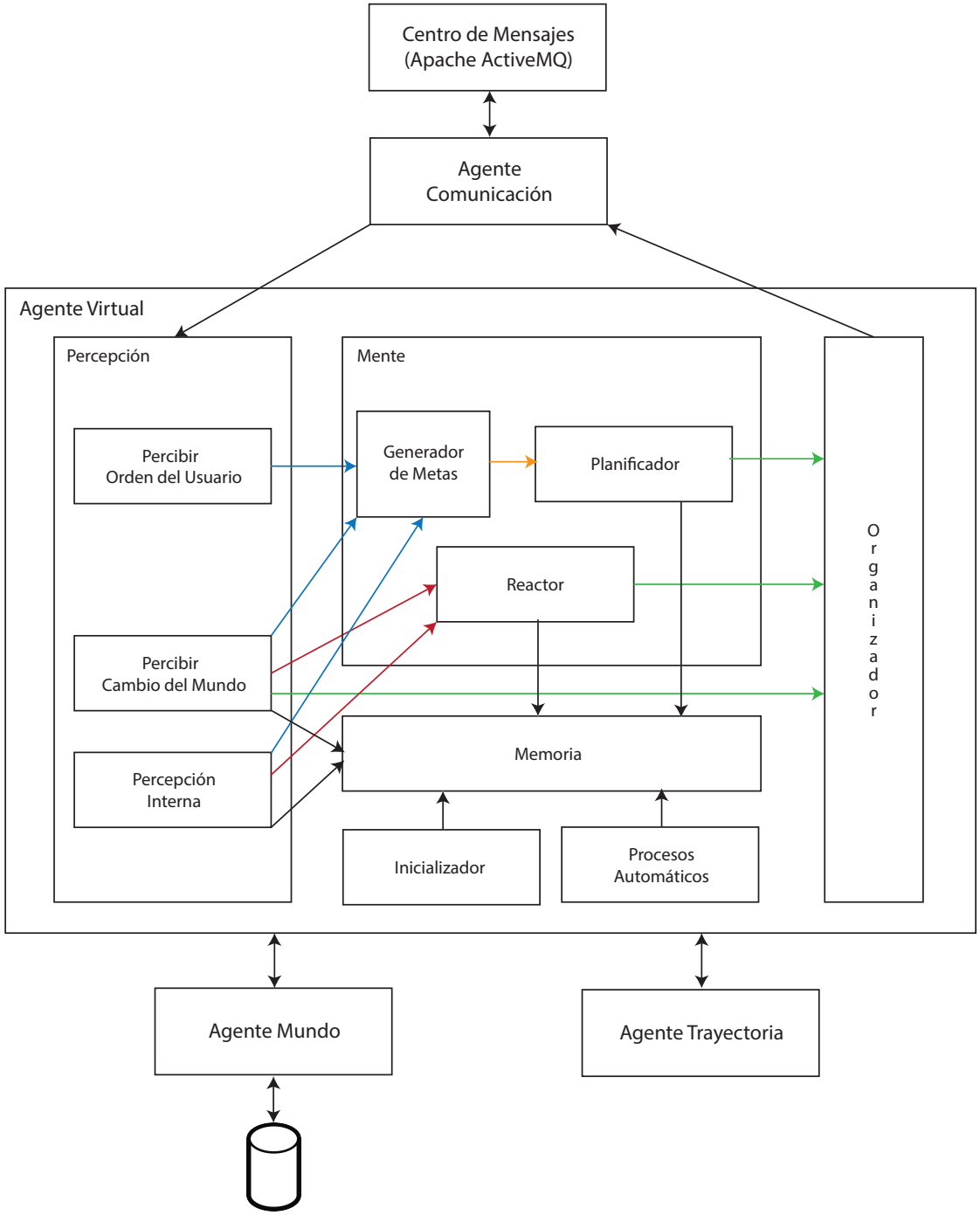

Figura 10: Arquitectura del Agente Virtual

Esta arquitectura, que representa al Agente Virtual, es sin duda la clave del sistema. Se puede destacar que son necesarias tres integraciones con otros agentes de la plataforma para ofrecer la funcionalidad completa, descrita en apartados anteriores. La arquitectura consta de diez módulos:

El **inicializador** proporciona las creencias y conocimientos con los que comienza el agente cuando se arranca el sistema.

La **memoria** es un módulo central que mantiene las creencias del agente sobre el mundo que le rodea y los conocimientos adquiridos durante su ciclo de vida. Gracias a estas creencias y conocimientos el agente va evolucionando y el sistema se vuelve dinámico.

Los **procesos automáticos** son procesos involuntarios propios del agente, tales como vivir o el consumo de energía que desemboca en la necesidad de comer o de descansar. Conforme se van ejecutando estos procesos, la memoria se va actualizando con el estado actual, permitiendo que otros módulos puedan consultarlo y actuar en consecuencia.

El módulo **percibir orden del usuario** recibe las percepciones que envía el Controlador, a través del Centro de Mensajes.

El módulo **percibir cambio del mundo** recibe las percepciones que envía el Cliente Gráfico, a través del Centro de Mensajes.

Por último, la **percepción interna** es la encargada de captar los cambios internos provocados por los procesos automáticos o la ejecución de acciones.

El **reactor** es el responsable de crear tareas simples, a partir de los estímulos que reciben los módulos de percepción.

El **generador de metas** se encarga de crear metas a partir de los estímulos que reciben los módulos de percepción.

El **planificador** es el encargado de generar planes, conjuntos de tareas que cumplen las metas fijadas por el generador de metas. Este módulo se integra con el Agente Mundo y el Agente Trayectoria, puesto que la mayoría de los planes necesitan conocer datos del mundo, y en caso de ser necesarios desplazamientos por el entorno, se requerirá calcular la trayectoria hasta el destino.

El **organizador** se encarga de ordenar, según su prioridad, las tareas generadas por el reactor y el planificador. Además, en caso de manejar planes se encarga de cumplir las restricciones de diseño impuestas. Por último, envía las tareas que el avatar tiene que ejecutar en el entorno virtual en el momento adecuado.

3.5.- Concreción Contextual

La arquitectura realizada es genérica, lo que permite que se puedan diseñar agentes virtuales con distintos tipos de percepciones y comportamientos. Este apartado está enfocado a describir las percepciones, tareas y comportamientos establecidos como ejemplo de Agente Virtual.

3.5.1.- Percepciones

A continuación, se describen las nueve percepciones realizadas a modo de ejemplo. En cada tabla, se especifica el nombre de la percepción, la precondición necesaria para capturarla, los metadatos que la complementan y el tipo de percepción que es.

Percepción	PERCEIVED_AVATAR.
Precondición	El avatar del Agente Virtual se encuentra delante de otro avatar en el entorno virtual.
Metadatos	Identificador único del avatar percibido.
Tipo	Percepción del mundo.

Percepción	DAYLIGHT.
Precondición	Se producen cambios de intensidad de la luz natural en el recinto donde se encuentra el avatar.
Metadatos	Intensidad: • FULL_LIGHT. • MEDIUM_LIGHT_INCREASING. • MEDIUM_LIGHT_DECREASING. • NO_LIGHT.
Tipo	Percepción del mundo.

Percepción	ARTIFICIAL_LIGHT.
Precondición	Se produce un cambio de estado de la luz artificial en el recinto donde se encuentra el avatar.
Metadatos	Estado: • Encendida (True). • Apagada (False).
Tipo	Percepción del mundo.

Percepción	LOW_BATTERY.
Precondición	El nivel de energía del Agente Virtual se encuentra al mínimo.
Metadatos	
Tipo	Percepción interna.

Percepción	FULL_BATTERY.
Precondición	El nivel de energía del Agente Virtual se encuentra al máximo.
Metadatos	
Tipo	Percepción interna.

Percepción	TASK_COMPLETED.
Precondición	Una tarea, que requiere de confirmación, ha finalizado.
Metadatos	Tipo de tarea completada.
Tipo	Percepción interna o del mundo. (Es una percepción que sirve de control al organizador).

Percepción	GOTO.
Precondición	El usuario especifica un destino.
Metadatos	Nombre del objeto junto al que se quiere ir.
Tipo	Orden del usuario.

Percepción	OPEN_DOOR.
Precondición	El usuario especifica una puerta.
Metadatos	Nombre de la puerta que se quiere abrir.
Tipo	Orden del usuario.

Percepción	CLOSE_DOOR.
Precondición	El usuario especifica una puerta.
Metadatos	Nombre de la puerta que se quiere cerrar.
Tipo	Orden del usuario.

Para tener una visión global de las percepciones se ha realizado la siguiente tabla, donde se muestra una síntesis de todas las anteriores y sus características:

Percepción	Metadatos	Precondición	Tipo
PERCEIVED_AVATAR	Id del avatar	Avatar en campo de visión	Mundo
DAYLIGHT	Intensidad	Cambio de intensidad	Mundo
ARTIFICIAL_LIGHT	Estado	Cambio de estado	Mundo
LOW_BATTERY		Nivel de energía al mínimo	Interna
FULL_BATTERY		Nivel de energía al máximo	Interna
TASK_COMPLETED	Tipo de Tarea	Finaliza una tarea	Mundo / Interna
GOTO	Nombre del objeto	Usuario especifica destino	Orden
OPEN_DOOR	Nombre de la puerta	Usuario especifica la puerta a abrir	Orden
CLOSE_DOOR	Nombre de la puerta	Usuario especifica la puerta a cerrar	Orden

3.5.2.- Tareas

En las próximas tablas se describen nueve tareas, que permiten llevar a cabo una serie de comportamientos de ejemplo. En cada tabla se especifica el nombre de la tarea, la descripción de su funcionalidad, los metadatos que la complementan, el tipo de tarea que es y su prioridad.

Tarea	GREET.
Descripción	El agente saluda al avatar captado en el entorno, mediante la percepción PERCEIVED_AVATAR.
Metadatos	
Tipo	Reactiva.
Prioridad	Inmediata.

Tarea	TURNON_LIGHT.
Descripción	El avatar enciende su linterna para iluminar el entorno cuando la intensidad de la luz natural (DAYLIGHT) no es suficiente y la linterna (ARTIFICIAL_LIGHT) se encuentra apagada.
Metadatos	
Tipo	Reactiva.
Prioridad	• Media si la intensidad es MEDIUM_LIGHT_DECREASING. • Alta si la intensidad es NO_LIGHT.

Tarea	TURNOFF_LIGHT.
Descripción	El avatar apaga su linterna cuando la intensidad de la luz natural (DAYLIGHT) es suficiente para iluminar el entorno y la linterna (ARTIFICIAL_LIGHT) se encuentra encendida.
Metadatos	
Tipo	Reactiva.
Prioridad	• Baja si la intensidad es MEDIUM_LIGHT_INCREASING. • Media si la intensidad es FULL_LIGHT.

Tarea	WALK.
Descripción	El avatar debe desplazarse a la posición especificada en los metadatos. Dicha posición siempre será accesible puesto que al realizar el plan se ha tenido en cuenta si en el camino había puertas cerradas.
Metadatos	Coordenadas 3D del destino.
Tipo	Plan.
Prioridad	Depende de la prioridad del plan.

Tarea	OPEN_DOOR.
Descripción	El avatar abre la puerta especificada en los metadatos.
Metadatos	Nombre de la puerta que se encuentra delante del avatar.
Tipo	Plan.
Prioridad	Depende de la prioridad del plan.

Tarea	CLOSE_DOOR.
Descripción	El avatar cierra la puerta especificada en los metadatos.
Metadatos	Nombre de la puerta que se encuentra delante del avatar.
Tipo	Plan.
Prioridad	Baja.

Tarea	START_CHARGE.
Descripción	El avatar se prepara para cargarse.
Metadatos	
Tipo	Plan.
Prioridad	Alta.

Tarea	CHARGING.
Descripción	El proceso de carga está ejecutándose.
Metadatos	
Tipo	Plan.
Prioridad	Alta.

Tarea	FINISH_CHARGE.
Descripción	El avatar termina de cargarse al recibir la percepción FULL_BATTERY. Y en este momento, se empiezan a aceptar nuevos planes puesto que el plan de carga no se puede cancelar.
Metadatos	
Tipo	Reactiva.
Prioridad	Inmediato.

Para tener una visión global de las tareas se ha realizado la siguiente tabla, que sintetiza toda la información de las anteriores y añade nuevos datos de interés para la comprensión del sistema. Como se puede observar, las tareas reactivas no necesitan confirmación de que han sido completadas, mientras que las tareas pertenecientes a planes sí la requieren. Además, las tareas reactivas no se pueden cancelar, mientras que para los planes dependerá del tipo que sea. Por último, señalar que los planes no pueden ser ejecutados con prioridad inmediata debido a que es un conjunto de tareas y es imposible ejecutar todas al mismo tiempo.

Tarea	**Metadatos**	**Tipo**	**Confirmación**	**Cancelable**	**Prioridad**
GREET	-	Reactiva	No	No	Inmediata
TURNON_LIGHT	-	Reactiva	No	No	Media/Alta
TURNOFF_LIGHT	-	Reactiva	No	No	Baja/Media
WALK	Coordenadas	Plan	Sí	Depende	Variable
OPEN_DOOR	Nombre puerta	Plan	Sí	Depende	Variable
CLOSE_DOOR	Nombre puerta	Plan	Sí	Sí	Baja
START_CHARGE	-	Plan	Sí	No	Alta
CHARGING	-	Plan	Sí	No	Alta
FINISH_CHARGE	-	Reactiva	No	No	Inmediato

3.5.3.- Comportamientos Reactivos

Una vez establecidas las percepciones y las tareas se puede formalizar, a través de las siguientes reglas, los comportamientos reactivos provocados por las percepciones:

1. PERCEIVED_AVATAR → GREET
   Interpretación: Si aparece un avatar en el campo de visión del agente, lo saludará.

2. DAYLIGHT = {MEDIUM_LIGHT_DECREASING | NO_LIGHT} &
   ARTIFICIAL_LIGHT = {FALSE} → TURNON_LIGHT
   Interpretación: Si está disminuyendo la intensidad de la luz natural y la luz artificial se encuentra apagada, se encenderá la luz artificial.

3. DAYLIGHT = {MEDIUM_LIGHT_INCREASING | FULL_LIGHT} &
   ARTIFICIAL_LIGHT = {TRUE} → TURNOFF_LIGHT
   Interpretación: Si esta aumentando la intensidad de la luz natural y la luz artificial se encuentra encendida, se apagará la luz artificial.

4. FULL_BATTERY → FINISH_CHARGE
   Interpretación: Si la batería alcanza su nivel de máxima energía, se finalizará el proceso de carga.

3.5.4.- Comportamientos Deliberativos

Los comportamientos deliberativos son mucho más complejos que los reactivos. Su formalización no es directa y es necesario hacerla mediante dos pasos. Primero, se generan metas a partir de las percepciones capturadas y, a continuación, se utilizan dichas metas para crear planes compuestos por varias tareas. La formalización del único comportamiento deliberativo definido en este proyecto se muestra dividido en los dos pasos anteriormente descritos.

Reglas de Generación de Metas

1. LOW_BATTERY → CHARGE

Reglas de Generación de Planes

1. CHARGE → (WALK OPEN_DOOR)* [WALK] START_CHARGE CHARGING
   Interpretación del plan: El avatar debe ir, caminando y abriendo las puertas, hasta la habitación que tiene el cargador. Una vez en la habitación, debe dirigirse al cargador. Situado encima de él, debe iniciar el proceso de carga.

3.5.5.- Comportamientos Semiautónomos

Los comportamientos semiautónomos siguen el mismo procedimiento que los deliberativos. Su formalización tampoco es directa y es necesaria realizarla con los mismos dos pasos. A continuación, se han formalizado tres comportamientos semiautónomos procedentes de órdenes del usuario.

Reglas de Generación de Metas

1. GOTO → IN_PLACED

2. OPEN_DOOR → DOOR_OPENED

3. CLOSE_DOOR → DOOR_CLOSED

Reglas de Generación de Planes

1. IN_PLACED → (WALK OPEN_DOOR)* [WALK]
   Interpretación del plan: El avatar debe ir, caminando y abriendo las puertas, hasta la habitación donde se encuentra su destino. Una vez en la habitación, debe dirigirse a la posición destino.

2. DOOR_OPENED → (WALK OPEN_DOOR)* [WALK] OPEN_DOOR
   Interpretación del plan: El avatar debe ir, caminando y abriendo las puertas, hasta la habitación donde se encuentra la puerta que tiene que abrir. Una vez en la habitación, debe dirigirse a la puerta y abrirla.

3. DOOR_CLOSED → (WALK OPEN_DOOR)* [WALK] CLOSE_DOOR
   Interpretación del plan: El avatar debe ir, caminando y abriendo las puertas, hasta la habitación donde se encuentra la puerta que tiene que cerrar. Una vez en la habitación, debe dirigirse a la puerta y cerrarla.

3.6.- Diseño Detallado

En este apartado se muestra el diseño detallado del sistema basándose en la arquitectura del Agente Virtual y la concreción contextual. En primer lugar, se presenta el diagrama de clases y, a continuación, los diagramas de secuencia de todas las percepciones.

3.6.1.- Diagrama de Clases

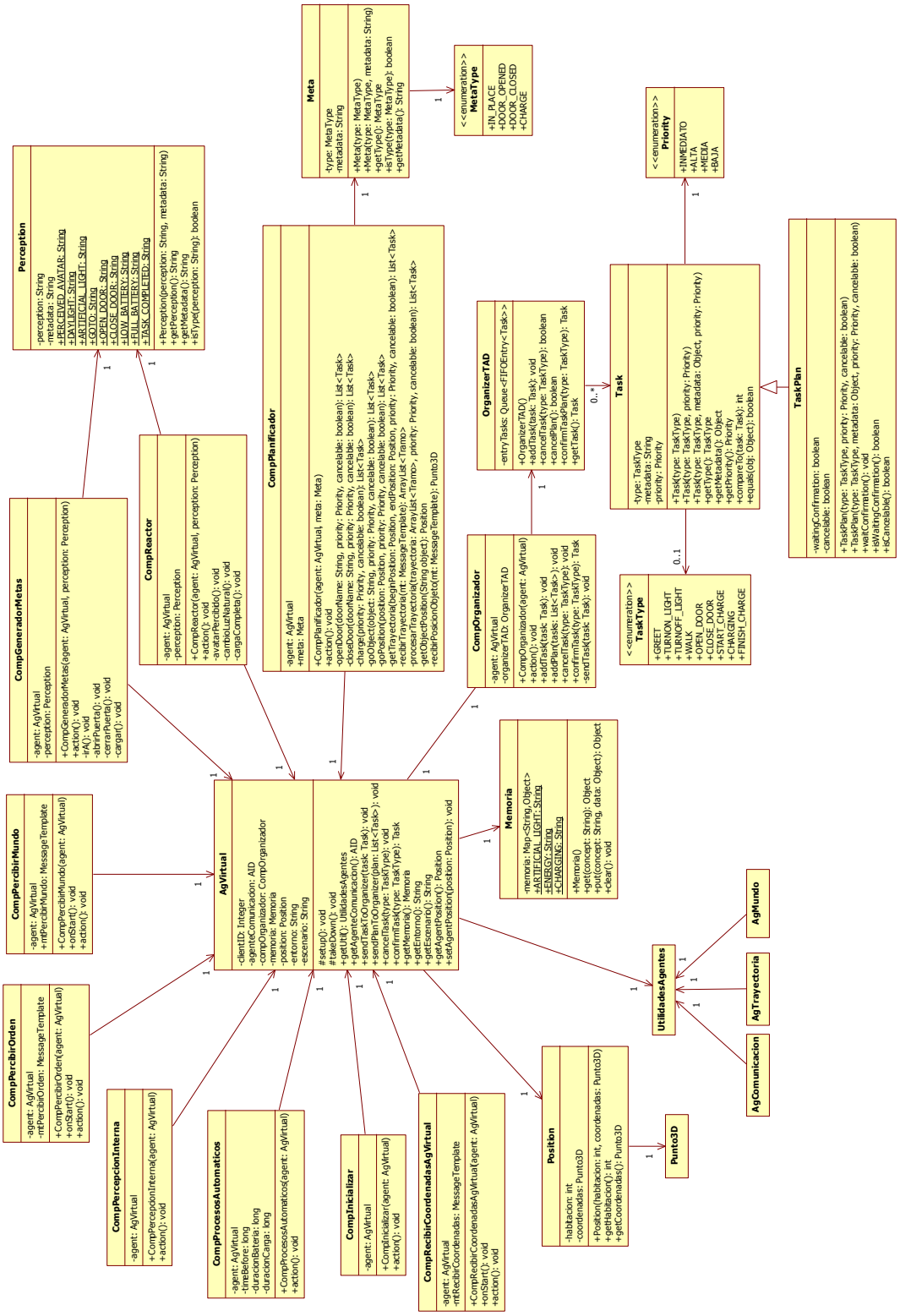

Figura 11: Diagrama de Clases del Agente Virtual

El diagrama muestra el diseño de clases del agente software. Se debe de tener en cuenta que se está haciendo un diseño orientado a agentes. En la figura, se observa una clase central que representa al propio agente. El resto de las clases son en su mayoría comportamientos.

De los diez comportamientos que tiene el agente, nueve de ellos representan a módulos de la arquitectura del Agente Virtual. El décimo, que no está representado en la arquitectura, se encarga de mantener sincronizada la posición del agente con la posición actual del avatar en el escenario.

La memoria ha sido diseñada como una clase independiente que pertenece al agente, y no como un comportamiento. De la misma forma han sido diseñadas las clases que permiten manejar las percepciones, tareas y metas.

Por último, se debe resaltar que el organizador dispone de una clase auxiliar, denominada OrganizadorTAD, que permite abstraer el complejo funcionamiento que presenta este componente.

3.6.2.- Diagramas de Secuencia

En este apartado se muestran los diagramas de secuencia, con distinto grado de abstracción, de todas las percepciones definidas en la concreción contextual. Seguidamente, se irán describiendo los diversos diagramas de cada percepción.

PERCEIVED_AVATAR

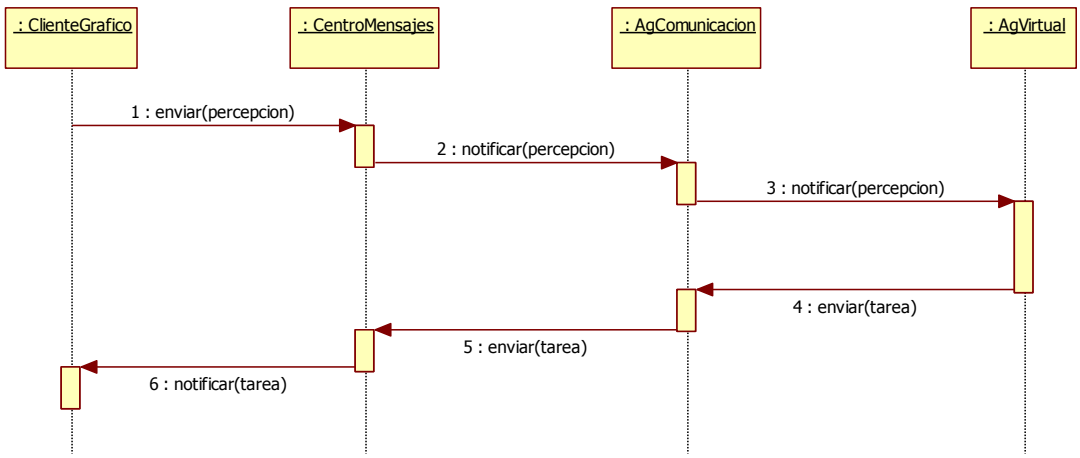

Figura 12: Diagrama de Secuencia General de PERCEIVED_AVATAR

En este diagrama se muestra que la percepción PERCEIVED_AVATAR es enviada desde el Cliente Gráfico hasta el Agente Virtual, que la procesa generando la tarea correspondiente (GREET) que el avatar ejecuta.

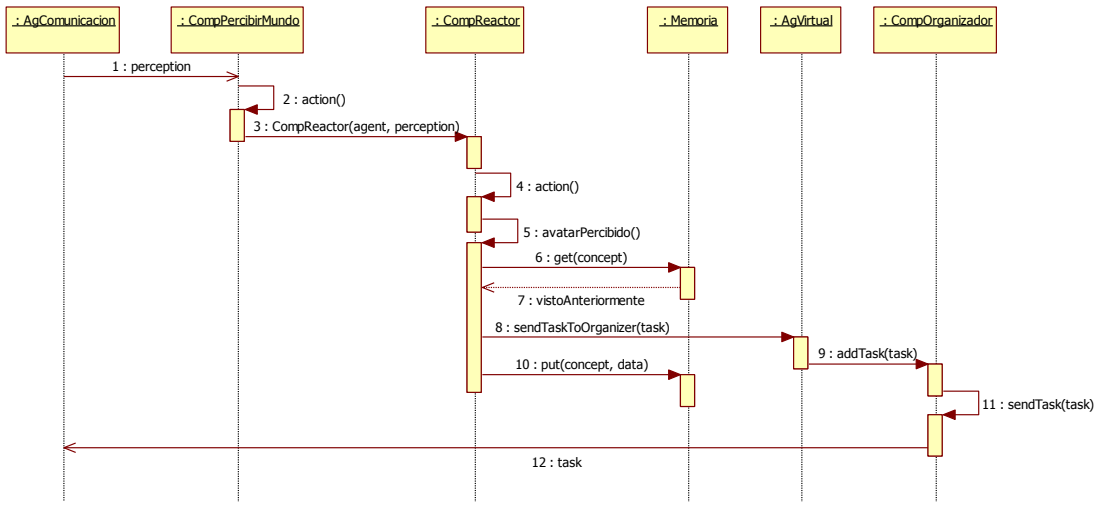

Figura 13: Diagrama de Secuencia Detallado de PERCEIVED_AVATAR

En esta figura muestra en detalle el flujo interno del Agente Virtual. La percepción PERCEIVED_AVATAR es recibida por el módulo que se encarga de percibir los cambios del mundo. Y éste se la pasa al reactor, ya que según su definición es reactiva. El reactor genera la tarea y se la envía al organizador que a su vez la envía al entorno virtual, cuando le corresponda según su prioridad.

DAYLIGHT

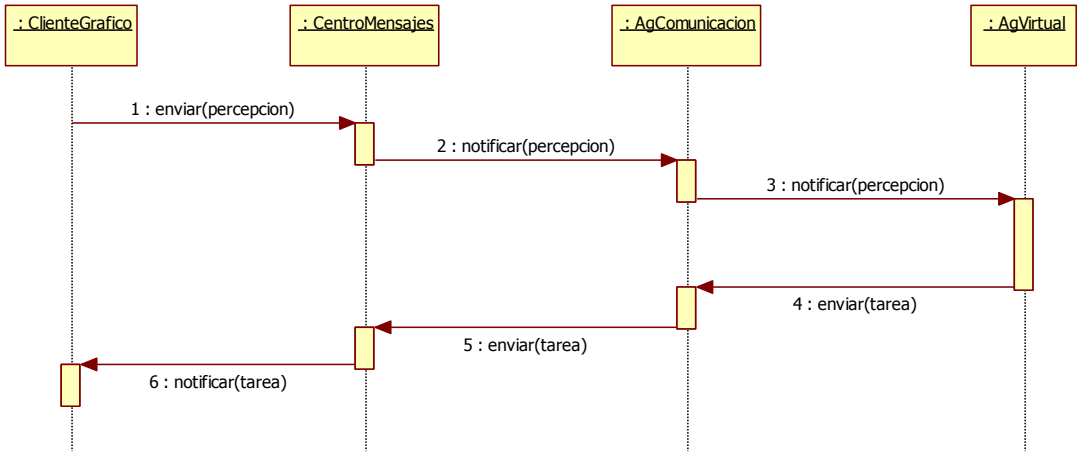

Figura 14: Diagrama de Secuencia General de DAYLIGHT

El diagrama anterior muestra que la percepción DAYLIGHT es enviada desde el Cliente Gráfico hasta el Agente Virtual, que la procesa generando la tarea correspondiente (TURNON_LIGHT o TURNOFF_LIGHT) que el avatar ejecuta.

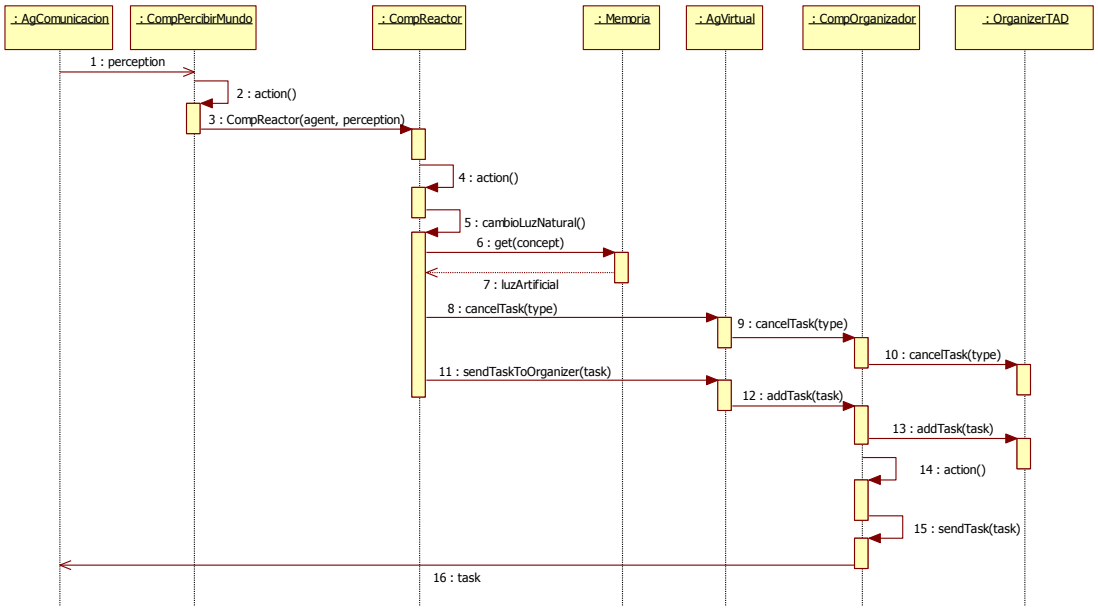

Figura 15: Diagrama de Secuencia Detallado de DAYLIGHT

La secuencia anterior muestra en detalle el flujo interno del Agente Virtual. La percepción DAYLIGHT es recibida por el módulo que se encarga de percibir los cambios del mundo. Y éste se la pasa al reactor, ya que según su definición es reactiva. El reactor comprueba las precondiciones y cancela todas las del mismo tipo que estuviesen pendientes. A continuación, genera una nueva tarea (TURNON_LIGHT o TURNOFF_LIGHT) y se la pasa al organizador que la envía al entorno virtual, cuando le corresponda según su prioridad.

ARTIFICIAL_LIGHT

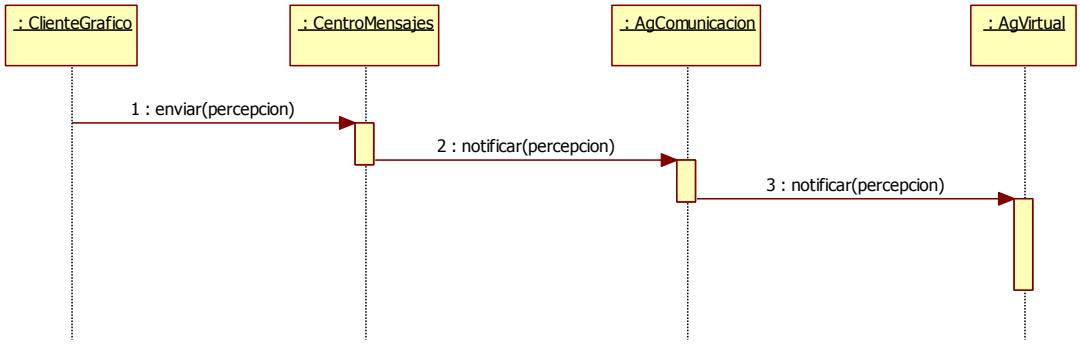

Figura 16: Diagrama de Secuencia General de ARTIFICIAL_LIGHT

En este diagrama se muestra que la percepción ARTIFICIAL_LIGHT es enviada desde el Cliente Gráfico hasta el Agente Virtual, que la procesa sin generar, en este caso, ninguna tarea que tenga que ser ejecutar.

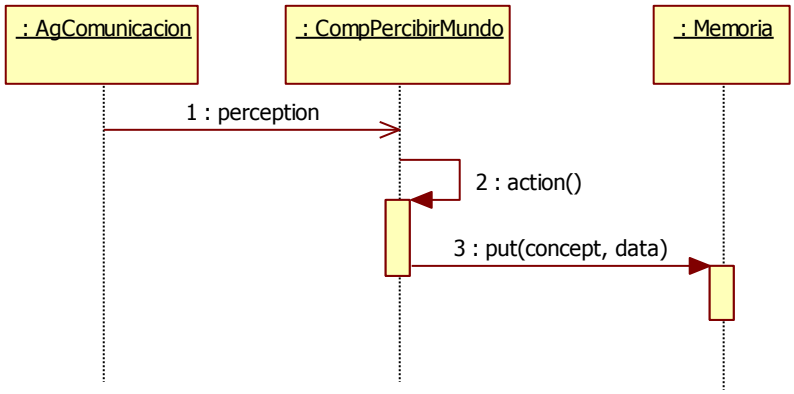

Figura 17: Diagrama de Secuencia Detallado de ARTIFICIAL_LIGHT

La figura anterior muestra en detalle el flujo interno del Agente Virtual. La percepción ARTIFICIAL_LIGHT es recibida por el módulo que se encarga de percibir los cambios del mundo. Este módulo actualiza el estado de la luz artificial en la memoria.

LOW_BATTERY

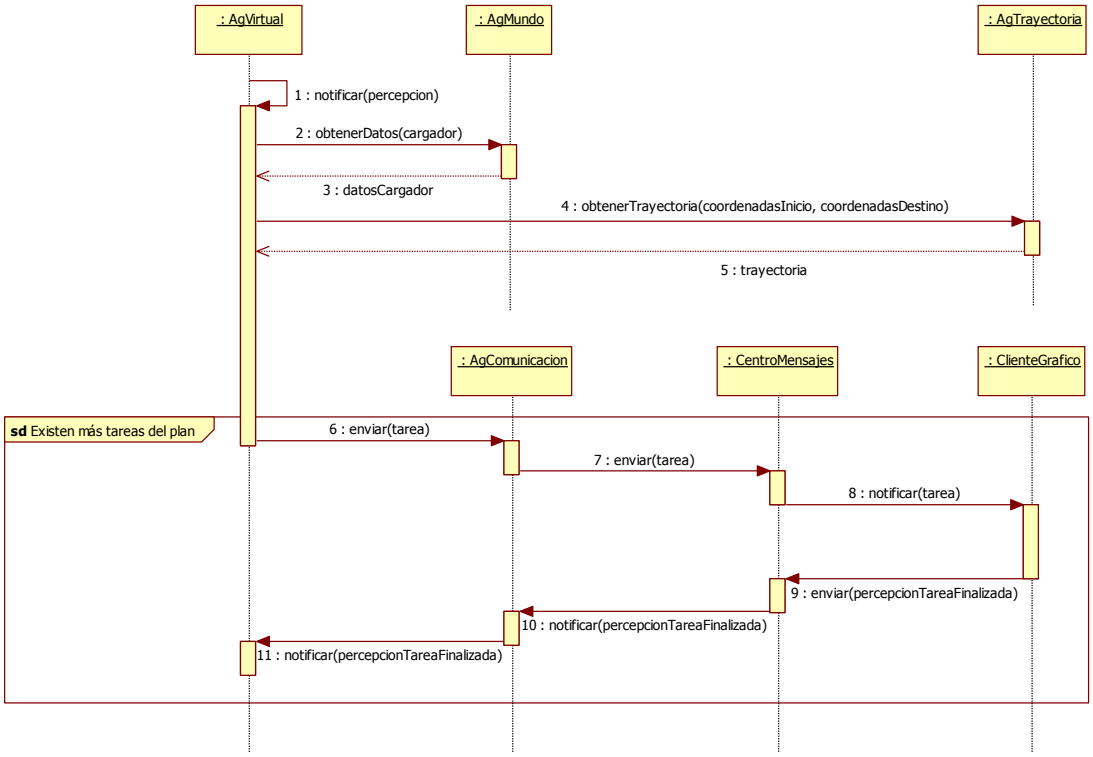

Figura 18: Diagrama de Secuencia General de LOW_BATTERY

En esta secuencia se muestra que la percepción LOW_BATTERY es captada internamente por el propio Agente Virtual. Éste la procesa, con ayuda del Agente Mundo y el Agente Trayectoria, y genera un plan con varias tareas que tienen que ser ejecutadas por el avatar y confirmadas una vez realizadas.

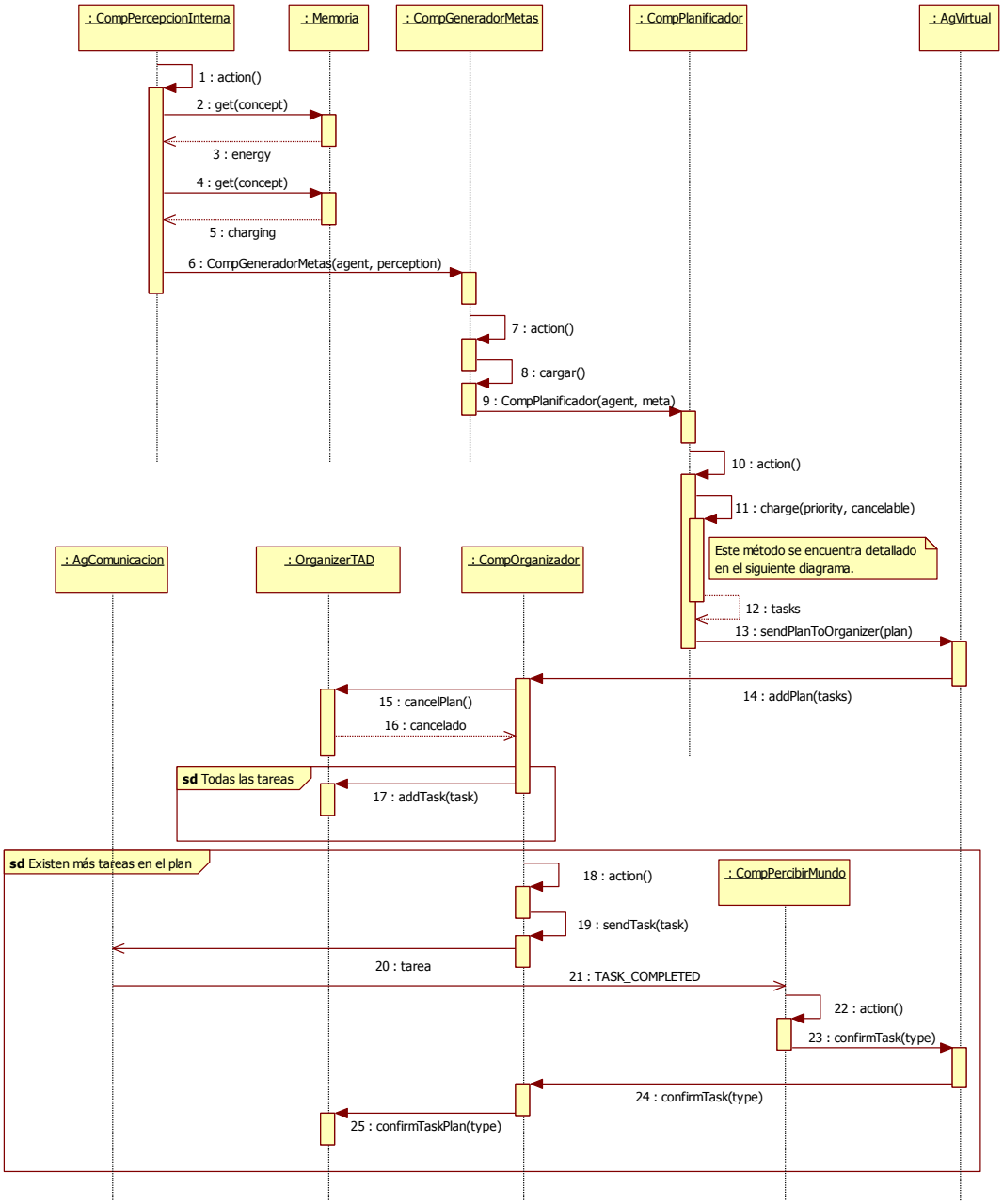

Figura 19: Diagrama de Secuencia Detallado de LOW_BATTERY

El diagrama anterior muestra en detalle el flujo interno del Agente Virtual. La percepción LOW_BATTERY es captada por el módulo de percepciones internas. Primero, se comprueba las precondiciones y, a continuación, se pasa la percepción al generador de metas. Éste crea la meta a la que se debe de llegar y se la envía al planificador, que se encarga de crear el conjunto de tareas para alcanzar dicha meta. El plan realizado es enviado al organizador que cancela el plan anterior y añade el actual. Posteriormente, el organizador envía las tareas al entorno y espera su confirmación.

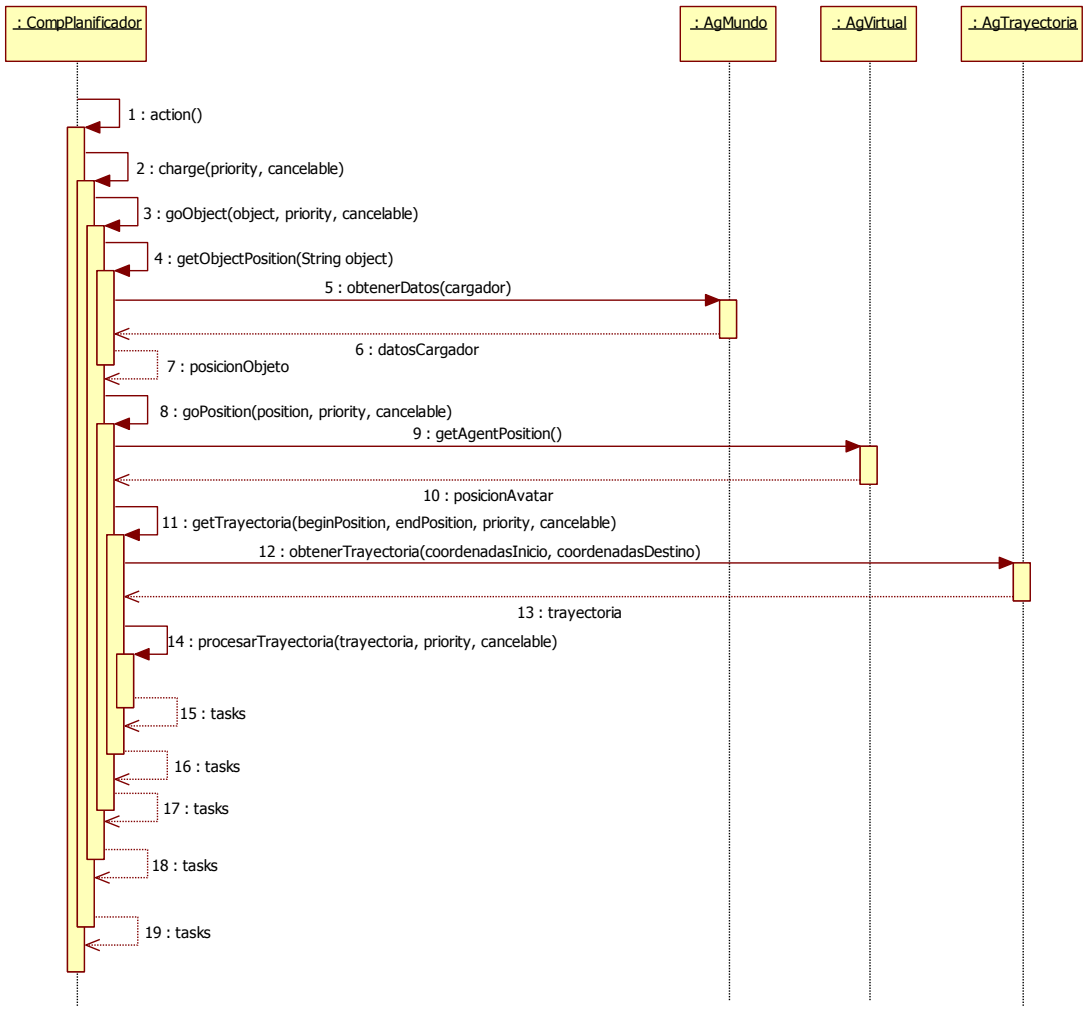

Figura 20: Diagrama de Secuencia del Planificador de LOW_BATTERY

La figura anterior muestra en detalle el funcionamiento del planificador. Uno de los requisitos era crear tareas de forma jerarquizada. El enfoque basado en jerarquías permite la reutilización pero complica la lectura, debido a que se debe ir profundizando en los comportamientos para obtener las tareas reales. En este caso, la meta es cargar completamente la batería. Dicha meta requiere, en primer lugar, ir hasta el cargador. Para ello, se debe preguntar al Agente Mundo la posición del cargador. Conociendo esta posición, hay que dirigirse hasta ella. Por tanto, se utiliza el Agente Trayectoria, que nos proporciona el punto inicial, las posiciones de las puertas por las que hay que ir pasando y el punto final destino. Esta trayectoria es procesada para convertirla en las tareas (WALK OPEN_DOOR)* [WALK]. De esta forma, disponemos de todas las tareas para situarnos encima del cargador. A continuación, se añaden las tareas START_CHARGE y CHARGING. Una vez se completen todas y cada una de las tareas del plan, se habría cumplido la meta.

FULL_BATTERY

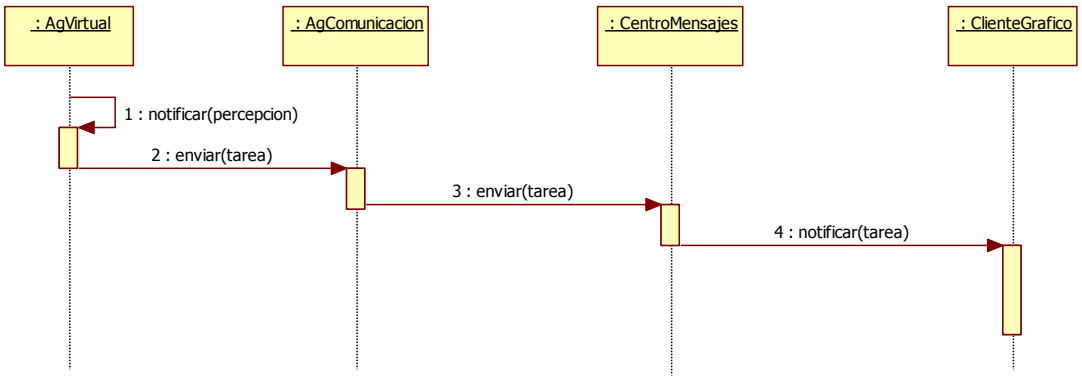

Figura 21: Diagrama de Secuencia General de FULL_BATTERY

El diagrama anterior muestra que la percepción FULL_BATTERY es captada internamente por el propio Agente Virtual, que la procesa generando la tarea correspondiente (FINISH_CHARGE) que el avatar ejecuta.

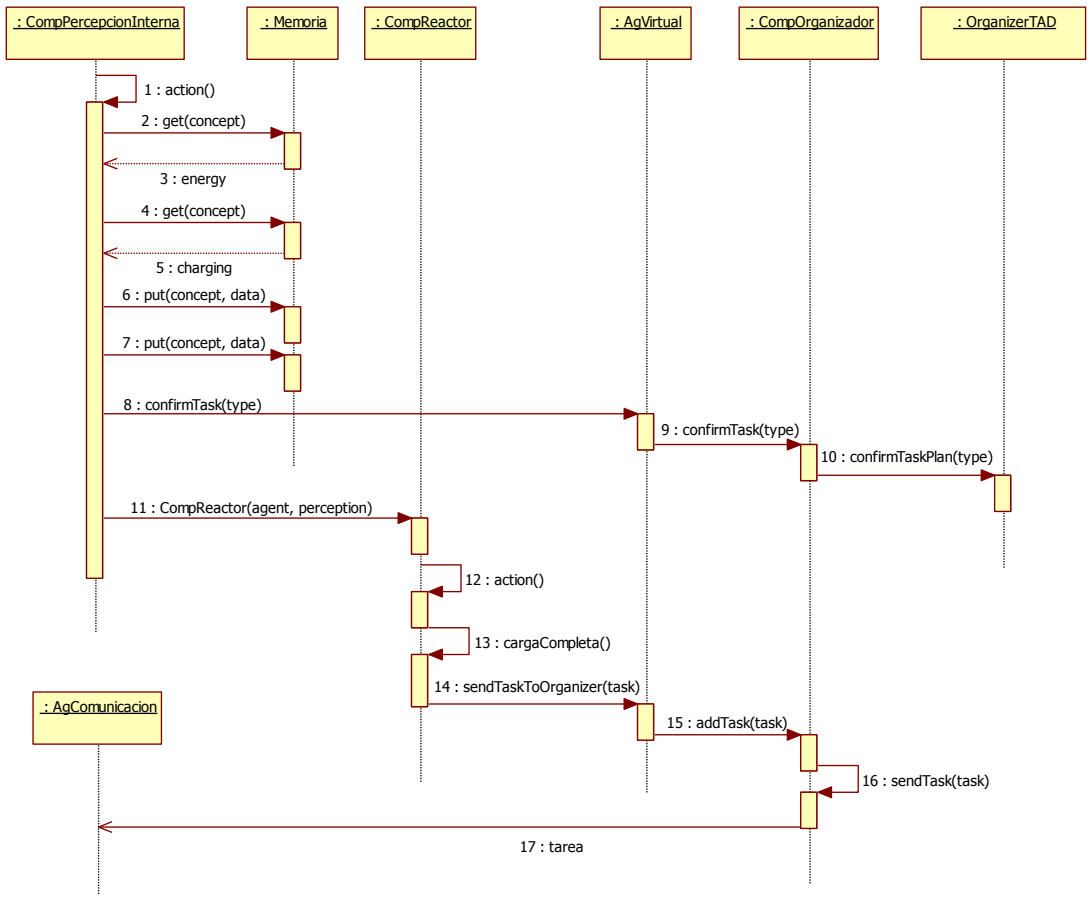

Figura 22: Diagrama de Secuencia Detallado de FULL_BATTERY

La figura anterior muestra en detalle el flujo interno del Agente Virtual. La percepción FULL_BATTERY es captada por el módulo de percepciones internas. Este módulo, en primer lugar, comprueba las precondiciones y confirma que se ha completado la tarea CHARGING. A continuación, se pasa la percepción al reactor, ya que según su definición es reactiva. Y éste genera la tarea correspondiente (FINISH_CHARGE) y se la pasa al organizador que a su vez la envía al entorno virtual, cuando le corresponda según su prioridad.

GOTO

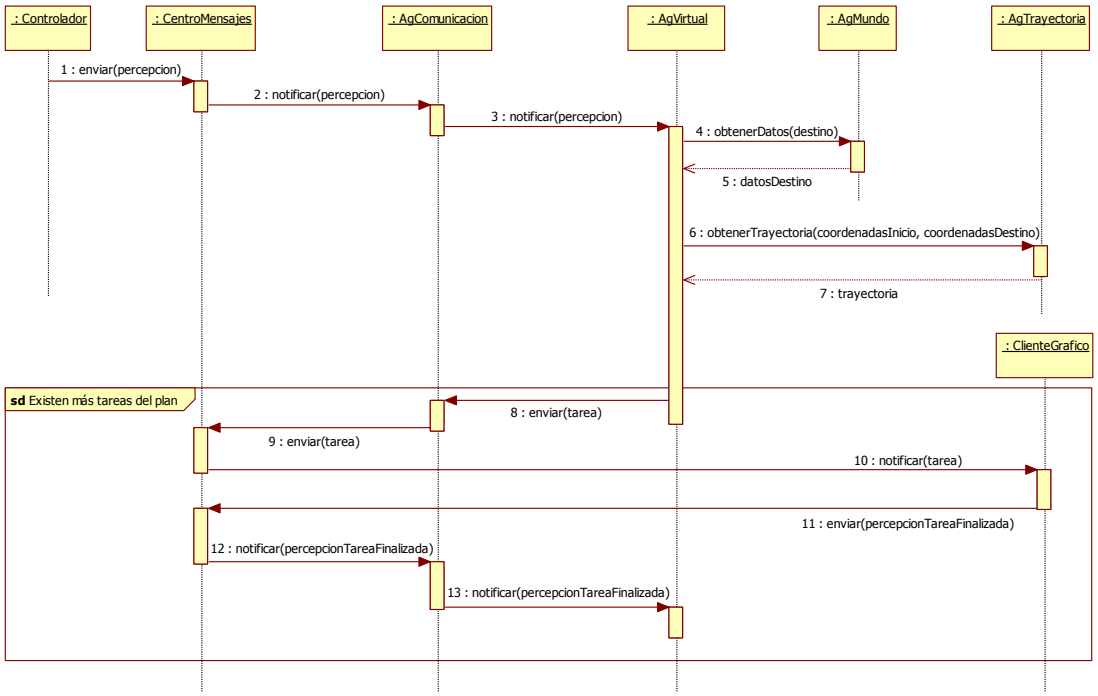

Figura 23: Diagrama de Secuencia General de GOTO

En esta secuencia se muestra que la percepción GOTO es enviada desde el Controlador hasta el Agente Virtual. Éste la procesa, con ayuda del Agente Mundo y el Agente Trayectoria, y genera un plan con varia tareas que tienen que ser ejecutadas por el avatar y confirmadas una vez realizadas.

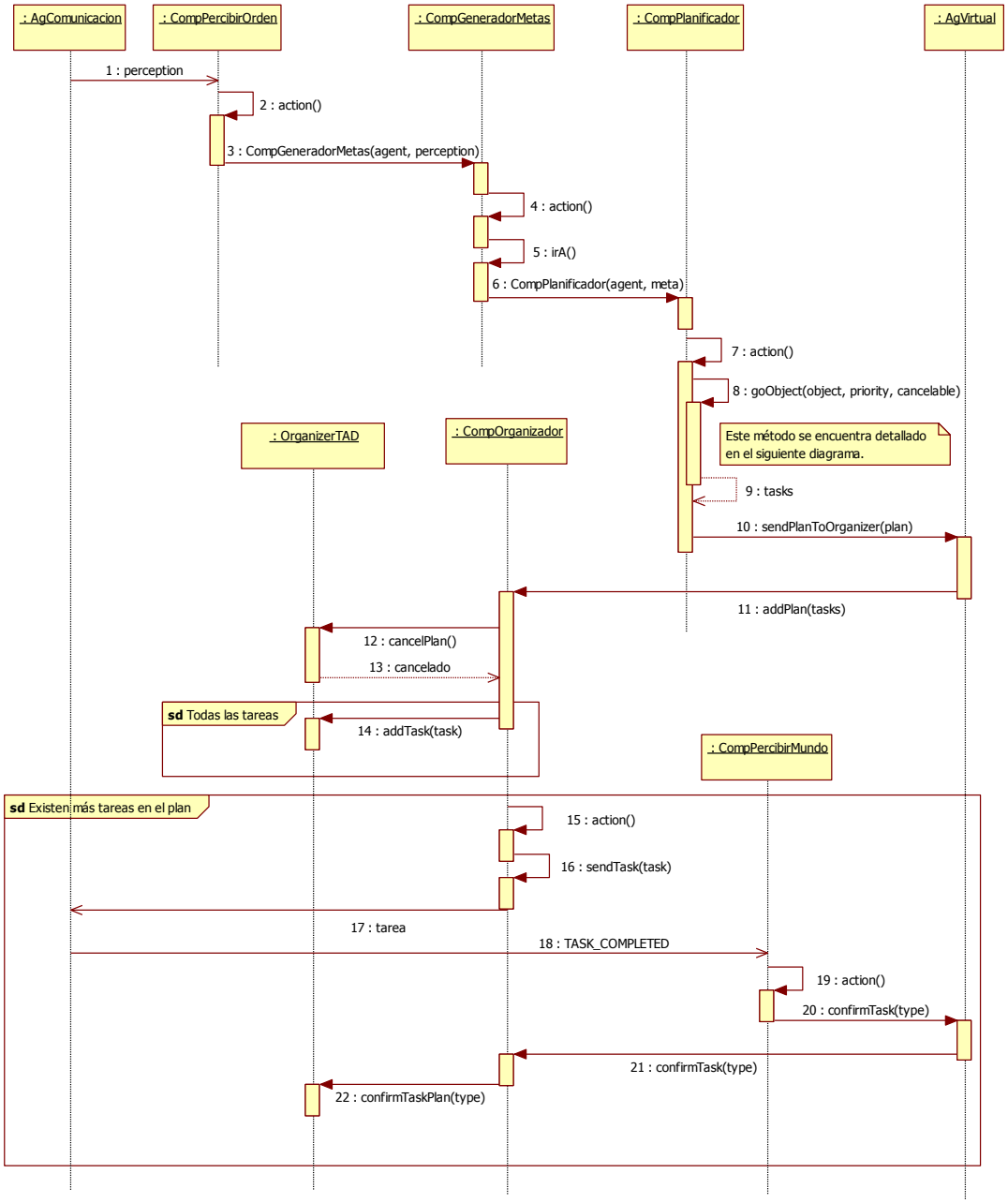

Figura 24: Diagrama de Secuencia Detallado de GOTO

El diagrama anterior muestra en detalle el flujo interno del Agente Virtual. La percepción GOTO es recibida por el módulo de percepciones de ordenes del usuario que la envía al generador de metas. Éste crea la meta a la que se debe llegar y se la envía al planificador, que se encarga de crear el conjunto de tareas para alcanzar dicha meta. El plan realizado es enviado al organizador que cancela el plan anterior y añade el actual. Posteriormente, el organizador envía las tareas al entorno y espera su confirmación.

Figura 25: Diagrama de Secuencia del Planificador de GOTO

La figura anterior muestra en detalle el funcionamiento del planificador. En este caso, la meta es llegar a un objeto o habitación destino. Para ello, se debe preguntar al Agente Mundo la posición del objeto o habitación destino. Conociendo esta posición, hay que dirigirse hasta ella. Por tanto, se utiliza el Agente Trayectoria, que proporciona el punto inicial, las posiciones de las puertas por las que hay que ir pasando y el punto final destino. Esta trayectoria es procesada para convertirla en las tareas (WALK OPEN_DOOR)* [WALK]. De esta forma, disponemos de todas las tareas necesarias para llegar hasta el destino. Por tanto, una vez se completen todas las tareas del plan, se habría cumplido la meta.

OPEN_DOOR

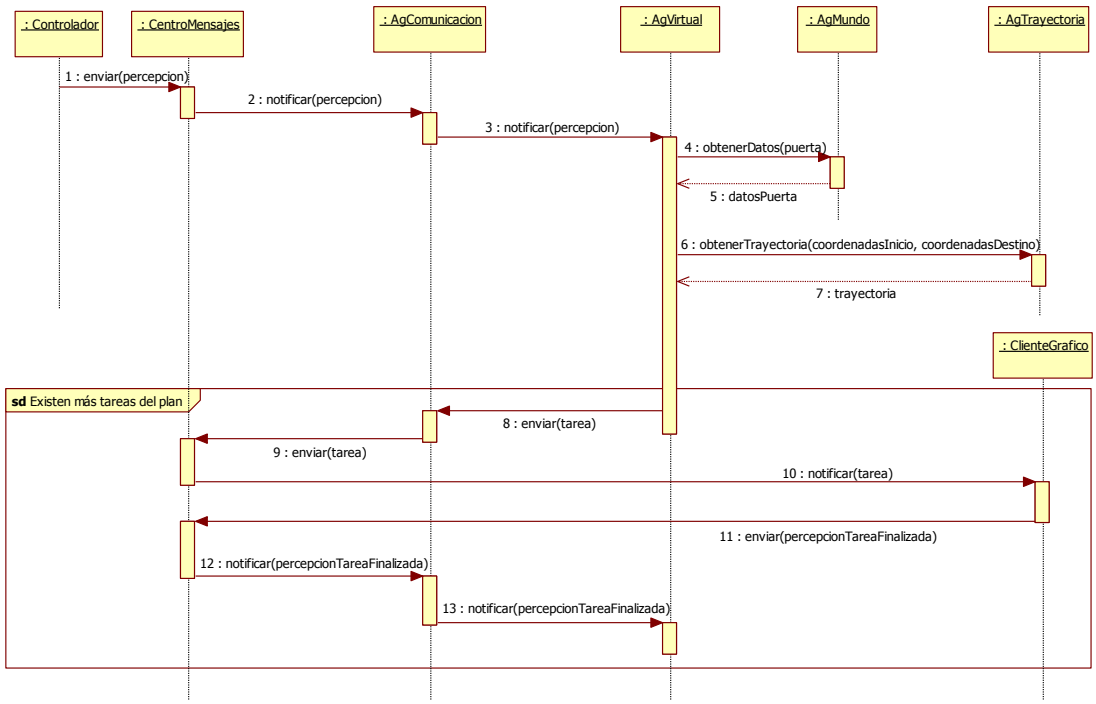

Figura 26: Diagrama de Secuencia General de OPEN_DOOR

En esta secuencia se muestra que la percepción OPEN_DOOR es enviada desde el Controlador hasta el Agente Virtual. Éste la procesa, con ayuda del Agente Mundo y el Agente Trayectoria, y genera un plan con varia tareas que tienen que ser ejecutadas por el avatar y confirmadas una vez realizadas.

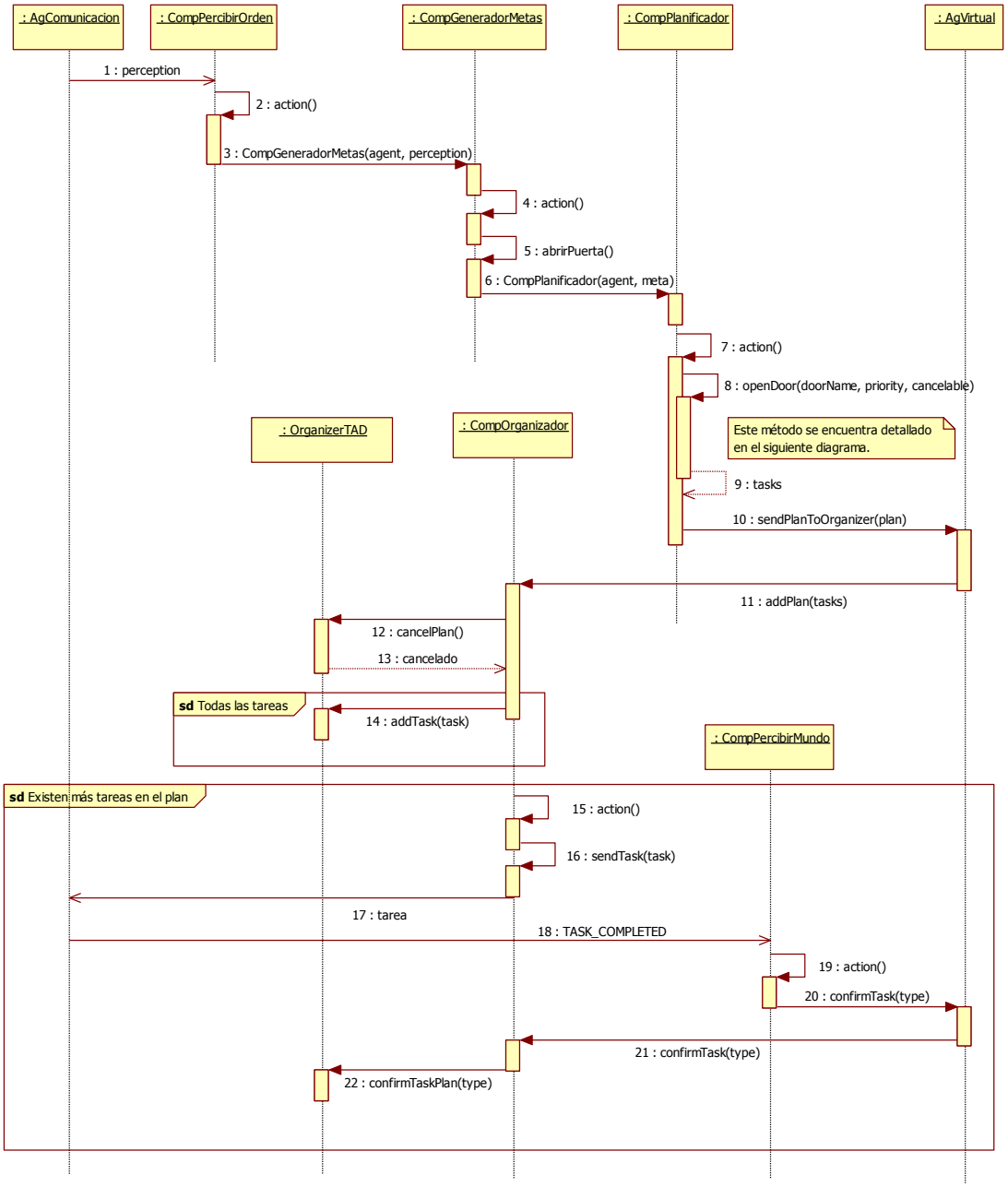

Figura 27: Diagrama de Secuencia Detallado de OPEN_DOOR

El diagrama anterior muestra en detalle el flujo interno del Agente Virtual. La percepción OPEN_DOOR es recibida por el módulo de percepciones de ordenes del usuario que la envía al generador de metas. Éste crea la meta a la que se debe llegar y se la envía al planificador, que se encarga de crear el conjunto de tareas para alcanzar dicha meta. El plan realizado es enviado al organizador que cancela el plan anterior y añade el actual. Posteriormente, el organizador envía las tareas al entorno y espera su confirmación.

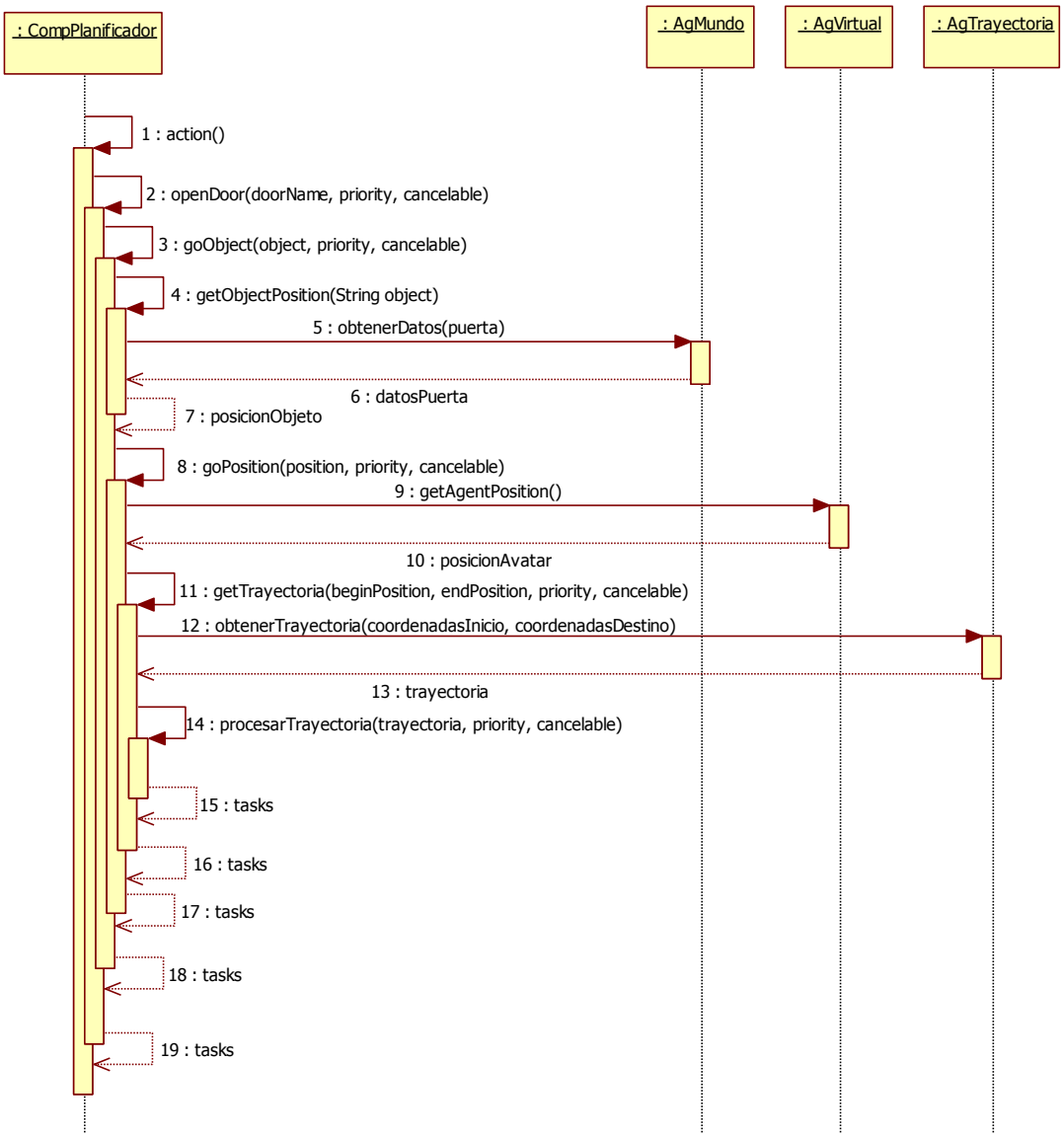

Figura 28: Diagrama de Secuencia del Planificador de OPEN_DOOR

La figura anterior muestra en detalle el funcionamiento del planificador. En este caso, la meta es abrir una puerta específica. Dicha meta requiere, en primer lugar, ir hasta la puerta. Para ello, se debe preguntar al Agente Mundo la posición de la puerta. Conociendo esta posición, hay que dirigirse hasta ella. Por tanto, se utiliza el Agente Trayectoria, que proporciona el punto inicial, las posiciones de las puertas por las que hay que ir pasando y el punto final destino. Esta trayectoria es procesada para convertirla en las tareas (WALK OPEN_DOOR)* [WALK]. De esta forma, disponemos de todas las tareas para situarnos delante de la puerta. A continuación, la única tarea que falta añadir es OPEN_DOOR. Una vez se completen todas las tareas del plan, se habría cumplido la meta.

CLOSE_DOOR

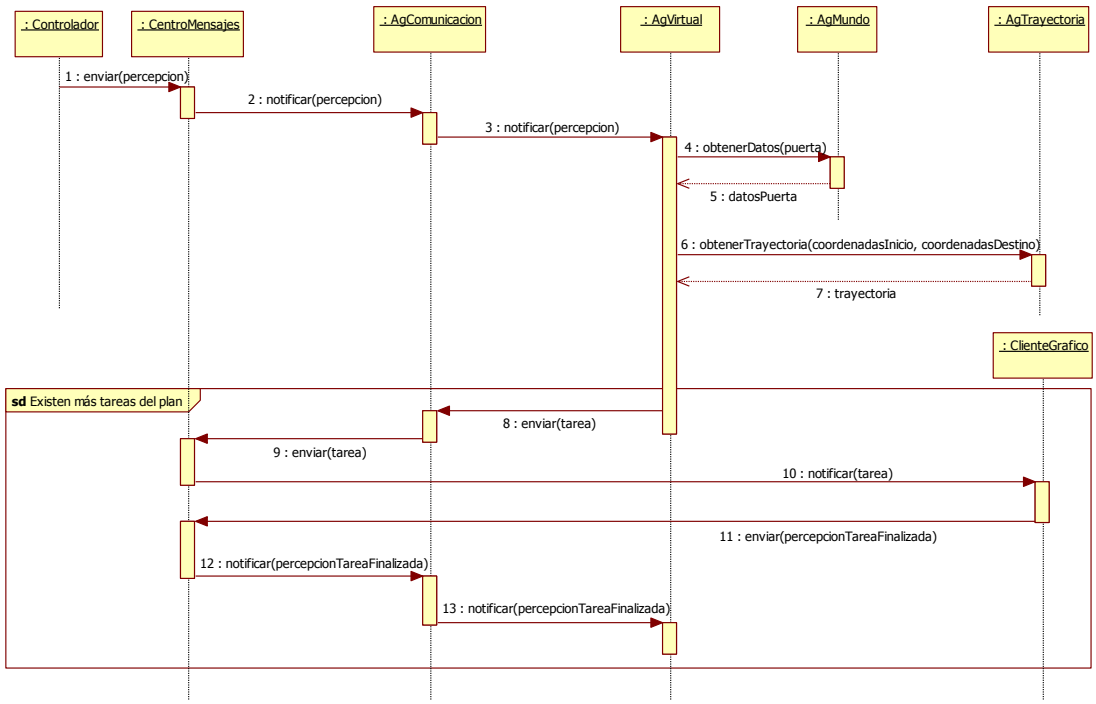

Figura 29: Diagrama de Secuencia General de CLOSE_DOOR

En esta secuencia se muestra que la percepción CLOSE_DOOR es enviada desde el Controlador hasta el Agente Virtual. Éste la procesa, con ayuda del Agente Mundo y el Agente Trayectoria, y genera un plan con varia tareas que tienen que ser ejecutadas por el avatar y confirmadas una vez realizadas.

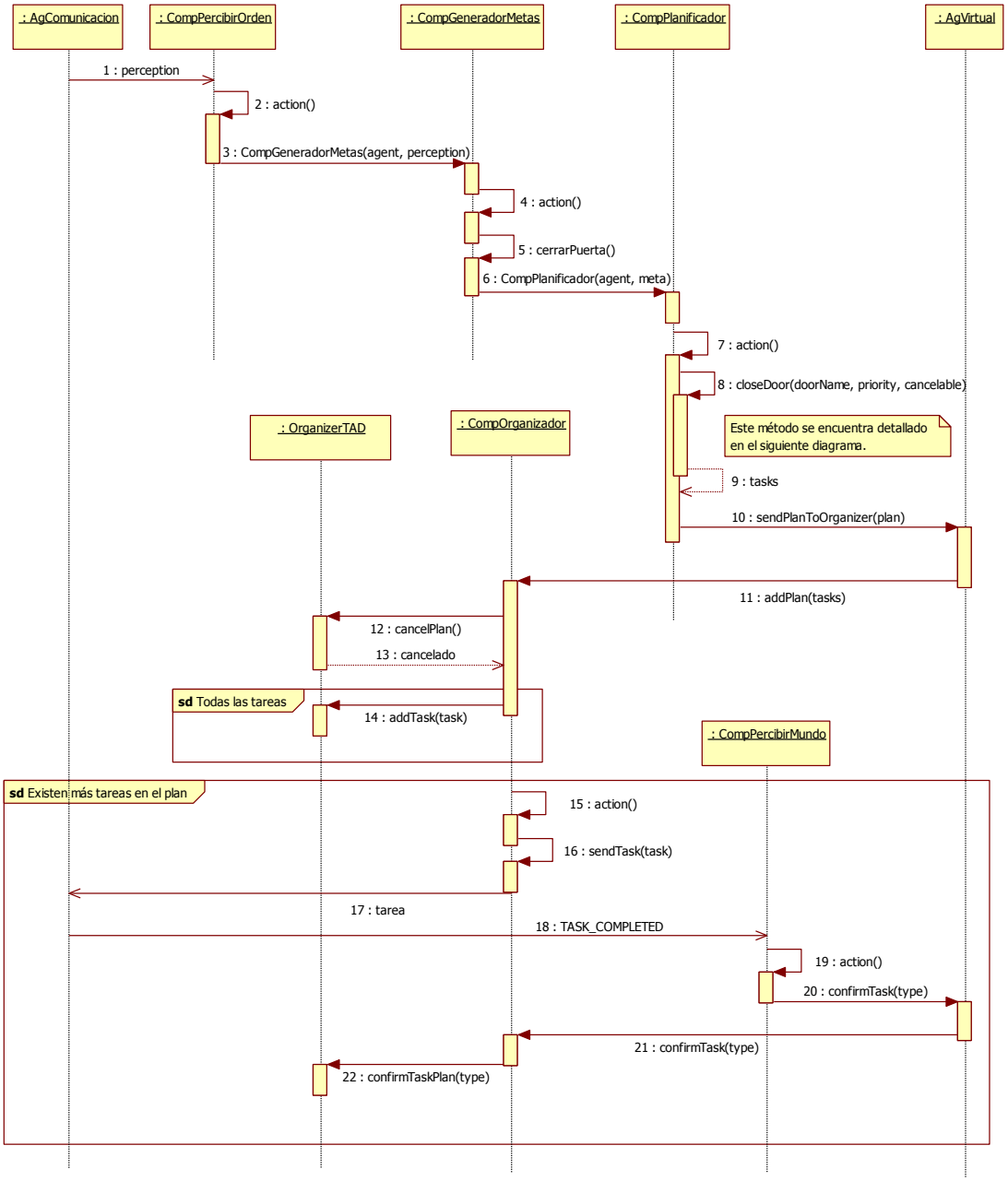

Figura 30: Diagrama de Secuencia Detallado de CLOSE_DOOR

El diagrama anterior muestra en detalle el flujo interno del Agente Virtual. La percepción CLOSE_DOOR es recibida por el módulo de percepciones de ordenes del usuario que la envía al generador de metas. Éste crea la meta a la que se debe llegar y se la envía al planificador, que se encarga de crear el conjunto de tareas para alcanzar dicha meta. El plan realizado es enviado al organizador que cancela el plan anterior y añade el actual. Posteriormente, el organizador envía las tareas al entorno y espera su confirmación.

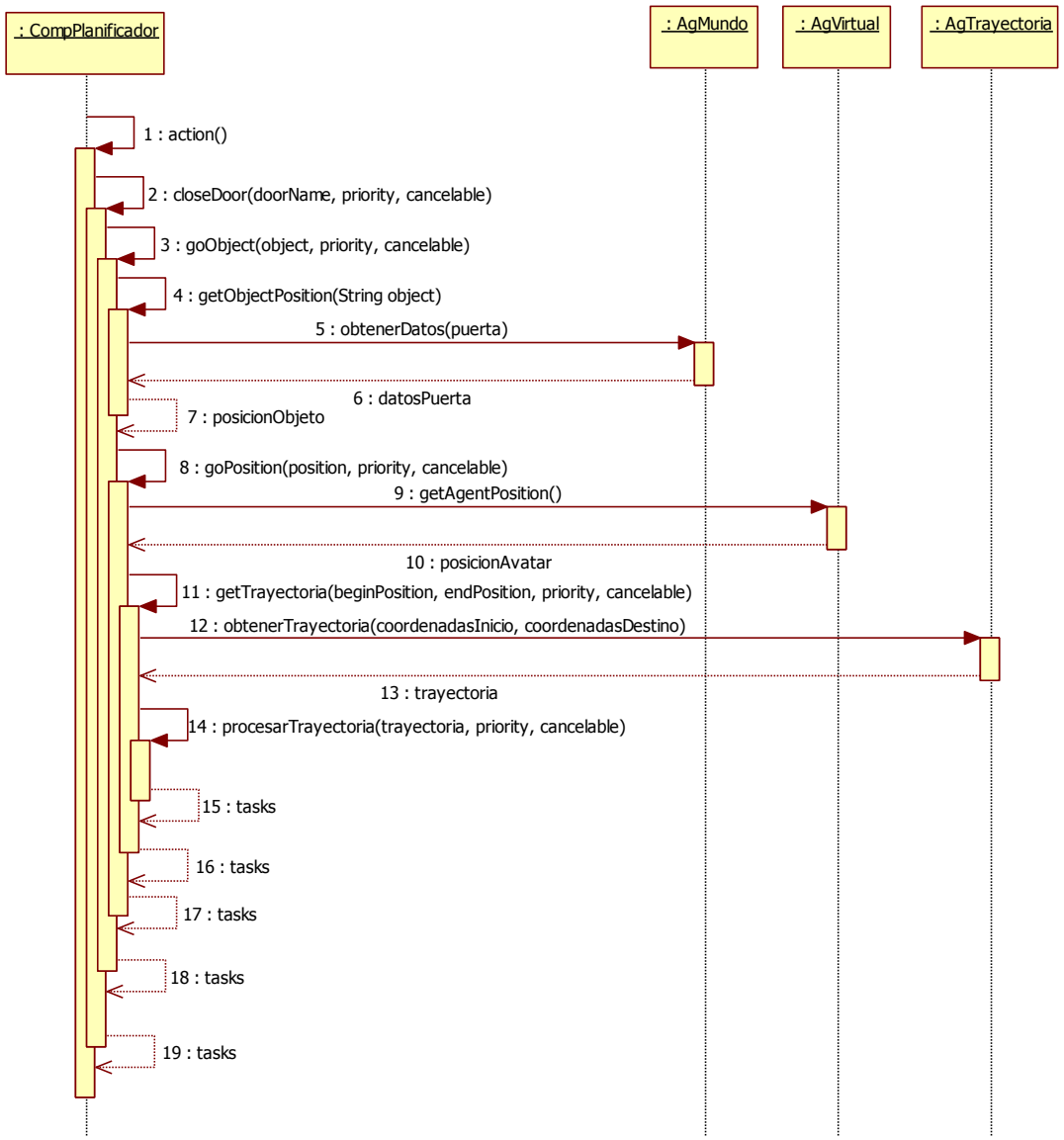

Figura 31: Diagrama de Secuencia del Planificador de CLOSE_DOOR

La figura anterior muestra en detalle el funcionamiento del planificador. En este caso, la meta es cerrar una puerta específica. Dicha meta requiere, en primer lugar, ir hasta la puerta. Para ello, se debe preguntar al Agente Mundo la posición de la puerta. Conociendo esta posición, hay que dirigirse hasta ella. Por tanto, se utiliza el Agente Trayectoria, que proporciona el punto inicial, las posiciones de las puertas por las que hay que ir pasando y el punto final destino. Esta trayectoria es procesada para convertirla en las tareas (WALK OPEN_DOOR)* [WALK]. De esta forma, disponemos de todas las tareas para situarnos delante de la puerta. A continuación, la única tarea que falta añadir es CLOSE_DOOR. Una vez se completen todas las tareas del plan, se habría cumplido la meta.

3.7.- Implementación

Este apartado detalla las herramientas utilizadas durante el desarrollo y ofrece las cifras relativas al tamaño del proyecto en cada una de sus partes. El método para conocer el número de líneas del código fuente ha consistido en contar el número total de líneas de los archivos sin tener en cuenta si son comentarios, llaves, líneas en blanco, etc.

Entorno Virtual

El entorno virtual fue realizado, como se ha comentado anteriormente, con Unity3D. La versión utilizada ha sido la 4.0, debido a que el avatar de pruebas necesitaba ciertas funcionalidades de esta versión. Este motor de videojuegos permite programar la lógica mediante scripts en C#, JavaScript y Boo. En este proyecto se ha decidido utilizar el lenguaje de programación C# porque que tiene tipado fuerte de datos, lo que minimiza los errores en tiempo de ejecución. Esta herramienta proporciona el IDE MonoDevelop para facilitar la programación de los scripts.

Los scripts programados se encuentran agrupados en subcarpetas dentro de la carpeta "Scripts", que se ubica en el espacio de trabajo del proyecto. El número de líneas del código programado para el entorno virtual son 2609.

Controlador

El Controlador es una aplicación desarrollada en Java. Y para facilitar su programación se ha utilizado el IDE Eclipse.

Esta aplicación esta distribuida en dos paquetes Java, uno para cada nivel de la arquitectura. La aplicación consta de 457 líneas.

Agente Virtual

El Agente Virtual es el sistema principal del proyecto. Ha sido desarrollado en Java siguiendo una enfoque orientado a agentes software. Para conseguir implementar agentes software se ha utilizado el framework JADE, en su versión 4.2.0, que proporciona interfaces para la creación de agentes de una manera rápida y sencilla. Además, se ha usado el IDE Eclipse para programar con mayor productividad.

La distribución del código sigue la estructura del proyecto MILES, puesto que el Agente Virtual se ha integrado dentro de su plataforma. El Agente Virtual se localiza en el paquete Java "agentes.virtual", donde se encuentran 25 clases agrupadas en varios subpaquetes. El diseño de clases, mostrado en apartados anteriores, presenta un número menor de clases debido a que en él no se han incorporado las clases auxiliares que ayudan a realizar una implementación más clara y sencilla. El número de líneas de todo el código del Agente Virtual asciende a 1373.

Por otra parte, el código que ha sido necesario para realizar la integración con la plataforma MILES se encuentra distribuido por distintos paquetes y clases de dicha plataforma. Por tanto, a los nombres de todos estos elementos se les ha añadido el sufijo "_AgVirtual" para conseguir una mejor organización y, además, poder relacionarlos con el Agente Virtual. El número de líneas para lograr esta integración ha sido de 784.

En resumen, el conjunto de las partes que forman este proyecto ha generado un sistema complejo con una configuración de 5223 líneas de código, distribuidas por todos sus componentes.

3.8.- Pruebas del Sistema

La realización de pruebas es un aspecto clave en el desarrollo de cualquier sistema software para comprobar su correcto funcionamiento y eliminar todos los errores que se detecten. En este proyecto, se han realizado pruebas funcionales, pruebas de integración y una depuración final del sistema.

Las pruebas funcionales tenían como objetivo buscar y corregir errores en el funcionamiento del Agente Virtual, del Controlador y del Cliente Gráfico. Se centraron en comprobar si la comunicación entre las distintas partes del sistema era la esperada. También, se revisó que cada percepción generase las tareas según las reglas definidas. Por último, se analizó si era correcto el orden de ejecución de varias tareas al mismo tiempo, según su prioridad y el cumplimiento de las restricciones de diseño impuestas. El método utilizado ha consistido en la división de las pruebas en las tres fases de desarrollo planificadas. En cada etapa se desarrolló un tipo de percepción que fue probada y corregida, si presentaba defectos, antes de continuar con la siguiente fase. Mediante estas pruebas se ha conseguido minimizar los errores, aunque no ha sido posible corregir ciertos errores, ajenos al proyecto, provocados por el funcionamiento del Agente Trayectoria en el cálculo de las mismas. Se puede concluir que estas pruebas han sido esenciales para conseguir un funcionamiento correcto del agente virtual casi en su totalidad.

Una vez finalizadas todas las pruebas funcionales y teniendo la certeza de que el agente virtual funcionaba correctamente, se realizó la integración con el robot arácnido. Para comprobar dicha integración se llevó a cabo las pruebas de integración. En ellas, se revisó que el avatar ejecutara los comportamientos adecuados que enviaba el Agente Virtual. Los resultados de estas pruebas no fueron completamente satisfactorios debido al modo de funcionamiento interno del robot arácnido y de su adaptación al entorno virtual. A pesar de todo, se ha conseguido visualizar los comportamientos generados por el Agente Virtual y demostrar, de esta manera, su correcto funcionamiento. De estas pruebas se puede concluir un resultado satisfactorio del Agente Virtual y la necesidad de una mejor integración del avatar con el entorno.

Finalmente, en la etapa de depuración del sistema, se han perfeccionado todas las partes del proyecto para lograr el resultado final obtenido. Entre las características obtenidas mediante esta depuración, se pueden destacar un aumento del realismo y una mayor fluidez en el sistema.

4.- RESULTADOS Y CONCLUSIONES

En este último capítulo se presentan los resultados obtenidos y las conclusiones alcanzadas durante el desarrollo del proyecto "Propuesta de integración de agentes software inteligentes sobre JADE con entornos virtuales en Unity3D". Además, se indican ciertas líneas de trabajo futuro que podrían ser seguidas para continuar mejorando y ampliando el Agente Virtual.

4.1.- Resultados

Las pruebas realizadas al sistema han proporcionado resultados positivos, así como un conjunto de aspectos que deberían ser revisados en la siguiente iteración de desarrollo.

Dentro de los aspectos positivos, se ha comprobado que el agente captura todas las percepciones definidas y realiza los comportamientos que derivan de ellas, según su prioridad y teniendo en cuenta las características propias de los mismos. Por esta razón, el primer y principal resultado que se debe destacar es el funcionamiento correcto y completo del agente según el diseño planteado.

El segundo resultado que se extrae es que no se aprovecha todo el potencial que ofrece el agente, para ello se debería diseñar un conjunto más amplio de comportamientos. Por ejemplo, sería interesante disponer de un comportamiento para utilizar los ascensores del edificio cuando haya que dirigirse a otra planta.

Por otra parte, el sistema responde sin retrasos significativos por lo que el usuario tiene sensación de fluidez. Bien es cierto que este resultado ha sido obtenido utilizando la red local del laboratorio y ordenadores con potentes GPUs que *renderizan* los gráficos tridimensionales en tiempo real. Aun así, se podrían realizar optimizaciones en los algoritmos utilizados para mejorar el tiempo de respuesta.

Uno de los resultados no alcanzados es una integración completamente satisfactoria con el avatar de prueba (robot arácnido). Esto es debido a una serie de factores entre los que se encuentran el modo de funcionamiento interno de dicho avatar, la forma de calcular el camino entre dos puntos y dificultades en la interacción con ciertos objetos del escenario, como son las puertas.

Otro resultado que se debe tener muy en cuenta es la extensibilidad conseguida en el sistema. Gracias a esta cualidad, se pueden añadir fácilmente nuevas funcionalidades a la arquitectura, por ejemplo para captar nuevos tipos de percepciones.

Además, teniendo en cuenta que la arquitectura presenta distintos módulos, nos permite reemplazar los componentes existentes por otros que ofrezcan mayor funcionalidad.

Por último, se ha detectado un comportamiento anómalo en el Agente Trayectoria (desarrollado por terceros) en el cálculo de ciertas trayectorias. Esto provoca un fallo crítico, en dicho agente, que impide realizar nuevos cálculos. Se puede destacar que el equipo encargado de su desarrollo ha sido informado del error, por lo que en un futuro próximo el funcionamiento será el correcto.

Respecto a los objetivos iniciales del proyecto, se puede destacar que se han cumplido de manera satisfactoria:

- El primer objetivo hacía referencia al desarrollo de una interfaz que permitiese la comunicación entre el entorno virtual y el agente software, como base para el desarrollo de un agente virtual inteligente. Este objetivo ha sido alcanzado gracias al desarrollo de los niveles de comunicación de la arquitectura y ha quedado demostrado en la fase de pruebas, porque las pruebas no se hubiesen podido llevar a cabo sin dicha interfaz.
- El segundo objetivo era establecer y programar un conjunto de comportamientos básicos. Durante las primeras fases del desarrollo se han definido los tipos de percepciones y de comportamientos que el agente debía tener. Posteriormente se ha realizado la concreción contextual para el avatar de pruebas, aunque la mayoría de los comportamientos especificados son independientes de la estructura corporal del avatar, como por ejemplo caminar hacia un punto de referencia.
- El tercer objetivo proponía el desarrollo de un agente de prueba. La concreción contextual constata este objetivo, puesto que define una serie de comportamientos a modo de ejemplo y, además, permite probar la arquitectura planteada.
- El cuarto objetivo planteaba la realización de pruebas al sistema. El apartado referente a pruebas en el capítulo de desarrollo y los resultados obtenidos demuestran que el objetivo se ha cumplido satisfactoriamente.
- El quinto objetivo, que perseguía documentar el desarrollo del proyecto, se ha alcanzado con la formalización de esta memoria.
- El último objetivo, documentar los mecanismos para la creación de nuevos agentes. Éste se podría haber cumplido de una manera más satisfactoria si se hubiese elaborado una guía breve enfocada exclusivamente a la programación de agentes virtuales, utilizando la base ya creada. A pesar de no tener esta guía, la lectura detallada del capítulo de desarrollo, de esta memoria, junto con el código programado del agente será suficiente para conseguir un desarrollo ágil. De nuevo, se debe reiterar que esto es debido a la modularidad y extensibilidad ofrecida por la arquitectura.

4.2.- Conclusiones

Los agentes virtuales inteligentes, en todas sus variantes, serán muy utilizados en el futuro debido a todas las aplicaciones que tienen en la vida real y a la diversidad de ámbitos donde pueden ser usados. El realismo de dichos agentes es superado día a día gracias al arduo trabajo llevado a cabo por investigadores, profesionales y empresas de todos los campos científicos, desde la psicología a las ciencias de la computación pasando por la biología o las matemáticas. Centrándonos en el campo de las Tecnologías de la Información, destacan los avances realizados en Inteligencia Artificial y en Ingeniería de Software durante las últimas décadas, lo que ha permitido alcanzar la madurez necesaria para crear nuevos sistemas complejos, siguiendo enfoques innovadores y evitando los errores del pasado.

Esta investigación sigue estas líneas de trabajo y abre la vía a nuevos enfoques y mejoras en el sistema. Este proyecto ha demostrado que la realización de agentes virtuales inteligentes mediante agentes software es una aproximación que se debe tener muy en cuenta en este tipo de desarrollos. Por ello, esta memoria y, en particular, la arquitectura planteada pueden ser de gran utilidad en trabajos futuros, puesto que proporcionan la base para un desarrollo organizado, extensible y modulado de todas las partes del sistema.

Aunque el proyecto se ha centrado mayoritariamente en el agente virtual, también se han dedicado recursos al desarrollo del entorno tridimensional, esto permite que personas sin conocimientos técnicos puedan comprender el trabajo realizado. Además, disponer de una visualización del sistema es un aspecto fundamental, porque ayuda a perfeccionar el funcionamiento del agente de una manera más sencilla, interactiva y en tiempo real.

El trabajo de integración de varios sistemas pone de manifiesto la dificultad que conlleva toda unificación de software. En muchos casos, dicha dificultad es provocada por no aplicar metodologías adecuadas de diseño y desarrollo para realizar sistemas más flexibles. Tenemos por delante la tarea de formarnos en metodologías actuales, que nos permitan realizar sistemas más complejos y perfeccionar la unión de los mismos para mejorar día a día su aplicación real. Apostar por la integración tecnológica es apostar por el progreso y desarrollo de la sociedad global en la que vivimos. A modo de ejemplo queda este agente virtual integrado en el proyecto MILES con el objetivo de explorar nuevas posibilidades de aplicar las Tecnologías de la Información y las Comunicaciones a la enseñanza y el aprendizaje en entornos virtuales tridimensionales.

La **conclusión final** que se puede extraer es que el presente trabajo ha abordado con éxito la integración y desarrollo de un agente virtual, aportando nuevas ideas y enfoques para futuras líneas de investigación.

4.3.- Trabajo Futuro

En este último apartado, se describen brevemente una serie de posibles líneas de trabajo futuro. Comenzaremos con las relacionadas directamente con este proyecto para concluir con alguna idea general.

Distinguir avatares: Si se diese el caso de que hubiera varios avatares dentro del campo de visión, actualmente se saludaría a todos ellos sin indicar qué avatar está siendo saludado específicamente. Una forma de perfeccionar el sistema es distinguir los avatares, lo que permitiría realizar comportamientos más personalizados.

Aumentar la expresividad de las percepciones y tareas: Las estructuras planteadas para las percepciones y las tareas podrían ser ampliadas y de esta forma permitir una definición más precisa, compleja y completa de las mismas. Por ejemplo, con las estructuras actuales y con un avatar humanoide, se podría saludar a otro avatar mediante un gesto con la mano. Si ampliamos la expresividad, podríamos conseguir especificar con qué mano saludar e incluso la fórmula oral que podría emplearse.

Mejorar la integración del avatar en el entorno virtual: El avatar no se encuentra integrado de forma óptima en el entorno virtual. Prueba de ello son, por ejemplo, los siguientes problemas: (1) cómo guardar una distancia de seguridad para poder abrir y cerrar las puertas sin que interfieran con el avatar, (2) cómo especificar de forma precisa que el avatar ha llegado a su destino o (3) cómo calcular un camino que evite los obstáculos, dejando un margen para no colisionar con ellos. Si se evitasen los problemas anteriores, se obtendría un mayor realismo en el entorno.

Eliminar las restricciones de diseño impuestas: Durante el desarrollo, se han establecido ciertos límites para simplificar el sistema. Por ejemplo, el agente solamente puede manejar un plan al mismo tiempo. Aunque para aumentar la flexibilidad del sistema, se permite cancelar los planes actuales para ejecutar otros diferentes. Estas restricciones podrían eliminarse con un nuevo organizador, más flexible, que permita manejar varios planes al mismo tiempo.

Crear una amplia gama de comportamientos: En este momento, solamente hay definidos unos pocos comportamientos a modo de ejemplo. El siguiente paso lógico consiste en ampliar el número de comportamientos manteniendo una jerarquía, es decir, la capacidad de formar comportamientos complejos a partir de otros ya existentes. Algunos ejemplos de posibles comportamientos serían encender la luz artificial pulsando interruptores, utilizar los ascensores o incluso algunos más abstractos como ir a trabajar, lo que implicaría una serie de subcomportamientos tales como dirigirse hasta la oficina, sentarse en el escritorio, encender el ordenador, etc.

Diseñar un planificador automático: Esta idea se encuentra relacionada con la anterior. Un planificador automático es un software que realiza planificaciones autónomamente gracias al uso de precondiciones, el estado actual del mundo virtual y otra serie de parámetros. Un planificador de este tipo, seguramente, proporcionaría al sistema mayor calidad y simplificaría la creación de nuevos comportamientos.

Aumentar el número y expresividad de las creencias: Esto se podría conseguir mediante el uso de una nueva memoria que permitiese almacenar nuevas creencias de mayor complejidad y riqueza.

Obtener estadísticas de uso y retroalimentación del estado actual: Actualmente, el Controlador es una aplicación independiente que solo permite enviar órdenes al usuario. Se puede aprovechar esta independencia para aumentar la funcionalidad de la aplicación, añadiendo elementos como información sobre el estado actual del agente, los comportamientos más utilizados, el estado del mundo virtual, etc.

Permitir la delegación de planes en otros agentes: En el estado actual de desarrollo el agente puede generar planes a partir de metas, pero la ejecución tiene que ser llevada a cabo por él mismo. Si diseñamos un entorno con varios agentes virtuales sería interesante que pudiesen cooperar para llevar a cabo un plan de forma conjunta o delegar el plan en otro agente.

Simulación de comportamientos animales: En la actualidad, la mayor parte del esfuerzo de distintos campos científicos se centra en la investigación del ser humano, seguramente el ser vivo más complejo conocido. En los entornos virtuales, es un error desarrollar únicamente comportamientos naturales para los seres humanos y no prestar atención a otros comportamientos, como los de los animales. El realismo final del entorno virtual estará limitado por el nivel de fidelidad del peor comportamiento simulado. Por ello, se sugiere esta línea para futuras investigaciones.

BIBLIOGRAFÍA

Aguilar, Virginia (2003). *Aplicación de Jack 2.3 al diseño del módulo de animaciones del proyecto VRIMOR*. (Trabajo Fin de Carrera). Facultad de Informática. Universidad Politécnica de Madrid. Madrid.

Badler, Norman (1997). *Real-time Virtual Humans*. Proc. IEEE 5th Pacific Conference on Computer Graphics and Applications.

De Antonio, Angélica; Ramírez, Jaime; Méndez, Gonzalo (2005). *An Agent-Based Architecture for Virtual Environments for Training.* En "Developing Future Interactive Systems", M.I. Sánchez-Segura (Ed.), pp. 212-233, 2 ISBN 1-59140-411-8, Idea Group Publishing.

De Antonio, Angélica; Imbert, Ricardo; Ramírez; Jaime; Ferré, Xavier (2004). *Usability issues in the Design of an Intuitive Interface for Planning and Simulating Maintenance Interventions using a Virtual Environment.* Virtual Reality, 7(3-4): 212-221. Special Issue on Interaction and Usability Issues for Desktop Virtual Environments.

Franklin, Stan; Graesser, Art (1996). Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. *En Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96),* tomo 1193. Springer-Verlag, Berlin, Germany.

Gratch, Jonathan; Rickel, Jeff; André, Elisabeth; Badler, Norman; Cassell, Justine; Petajan, Eric (2002). *Creating interactive virtual humans: some assembly required*. Intelligent Systems. IEEE, vol.17, no.4, pp. 54- 63, doi: 10.1109/MIS.2002.1024753.

Imbert, Ricardo (2005). *Una Arquitectura Cognitiva Multinivel para Agentes con Comportamiento Influido por Características Individuales y Emociones, Propias y de Otros Agentes.* (Tesis doctoral). Facultad de Informática. Universidad Politécnica de Madrid. Madrid.

Wooldridge, Michael; Jennings, Nicholas R. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2): 115–152.

Wooldridge, Michael; Jennings, Nicholas R. (1998). Pitfalls of Agent-Oriented Development. En K. P. Sycara y M. Wooldridge (eds.), *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pp. 385–391. ACM Press, New York.

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Fri Feb 14 18:55:26 CET 2014
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)